RAPID OBJECT DETECTION SYSTEMS, UTILISING DEEP LEARNING AND UNMANNED AERIAL SYSTEMS (UAS) FOR CIVIL ENGINEERING APPLICATIONS

David Griffiths*, Jan Boehm

UCL Department of Civil, Environmental & Geomatic Engineering, Gower Street, London, WC1E 6BT – (david.griffiths.16, j.boehm)@ucl.ac.uk

Commission II, WG II/6

KEY WORDS: Object detection, Deep Learning, Unmanned Aerial Systems, Railway, Rapid

ABSTRACT:

With deep learning approaches now out-performing traditional image processing techniques for image understanding, this paper accesses the potential of rapid generation of Convolutional Neural Networks (CNNs) for applied engineering purposes. Three CNNs are trained on 275 UAS-derived and freely available online images for object detection of $3m^2$ segments of railway track. These includes two models based on the Faster RCNN object detection algorithm (Resnet and Incpetion-Resnet) as well as the novel one-stage Focal Loss network architecture (Retinanet). Model performance was assessed with respect to three accuracy metrics. The first two consisted of Intersection over Union (IoU) with thresholds 0.5 and 0.1. The last assesses accuracy based on the proportion of track covered by object detection proposals against total track length. In under six hours of training (and two hours of manual labelling) the models detected 91.3%, 83.1% and 75.6% of track in the 500 test images acquired from the UAS survey Retinanet, Resnet and Inception-Resnet respectively. We then discuss the potential for such applications of such systems within the engineering field for a range of scenarios.

1. INTRODUCTION

With significant advances in the research field of deep learning, there has been a dramatic change in image processing techniques for image understanding and object recognition (LeCun *et al*., 2015). This has been driven by significant developments in the architectures of Convolutional Neural Networks (CNNs) (e.g. Long *et al*., 2014; Badrinarayanan *et al*., 2015; Chen *et al*., 2016). Such advances have led to object detection models trained using state-of-the-art CNNs to surpass human abilities to distinguish features within an image (Chen *et al*., 2015). Furthermore, adaptions of CNN object detection models are paving the way for accurate general-purpose pixel-wise segmentation and classification of images (Long *et al*., 2014; Pinheiro *et al*., 2016). Whilst CNNs are being heavily utilised within the computer science and medical imaging fields, there is less evidence of such exploitation within the field of engineering. There are many reasons why this may be the case, however, one of most obvious reasons would be the novelty of the technology. This has likely resulted in many engineering firms simply being unaware of the recent advances in both deep learning as well as the open-source libraries that allow for easy model generation and application. Here we demonstrate that with the aid of Googles TensorFlow open-source project as well as cloud computing resources (e.g. Amazon AWS, Google Cloud etc.), deep learning is accessible to a wide audience outside of the computer science and medical imaging field. To achieve this, we outline the methodology required to train a CNN using UAS acquired imagery from an aerial railway track survey. Furthermore, we address key obstacles in the generation of such models as well as undertake quantitative and qualitative comparisons of key decisions that must be made. These include model hyper-parameters such as training set size, learning rates and architectures. Whilst deep learning and CNN techniques have been well documented within computer science journals, this paper intends to demonstrate the potential of the technology in the context of practical application for engineering scenarios. The broader aim of this paper is to demonstrate the advantages and caveats of the use deep learning in this context.

1.1 Convolutional Neural Networks

CNNs (or ConvNets) are the most common deep learning tool for analysing visual imagery and since the success of Krizhevsky *et al*., (2012) architecture for the annual ImageNet classification competition (Deng *et al*., 2009) they have proved to be an essential tool for image understanding. The modern CNN architectures are largely accredited to pioneering work undertaken by LeCun *et al*. (1989) and LeCun *et al*. (1998). CNN's derive their name from the 'convolutional' operator which is a common tool used in image processing. Convolution filters have demonstrated essential components for extracting useful information from images, however, the convolution kernel needs to be pre-computed and pre-selected prior to being used on an image (e.g. Sobel, Laplacian, Gaussian blur filters etc.). By definition, deep learning does not follow this pattern. In deep learning, relationships between features are learned through supervised training of artificial neural networks. No prior assumptions or relationships are given to the model prior to training. Each neuron within the neural network is assigned a weight and bias which is used to manipulate the flow of input data. The weights and biases for each neuron are trained using a backpropagation algorithm (LeCun *et al*., 1990) which utilises supervised training data to calculate the necessary gradient and thus direction of each weight and bias to be tuned to minimise a loss function. Whilst a full review of deep learning technology is beyond the scope of this paper, LeCun *et al*. (2015) can serve for such a purpose. CNNs work in a similar way to typical deep learning networks, however, convolution filters are the features to be learned by the network. During training, images are passed through the CNN and the convolutions applied to the image. The filters are then adjusted accordingly through the backpropagation algorithm. While higher level filters tend to learn edge/corner detection filters, lower level filters can be significantly more complex and unique matrix combinations (Figure 1).

* Corresponding author

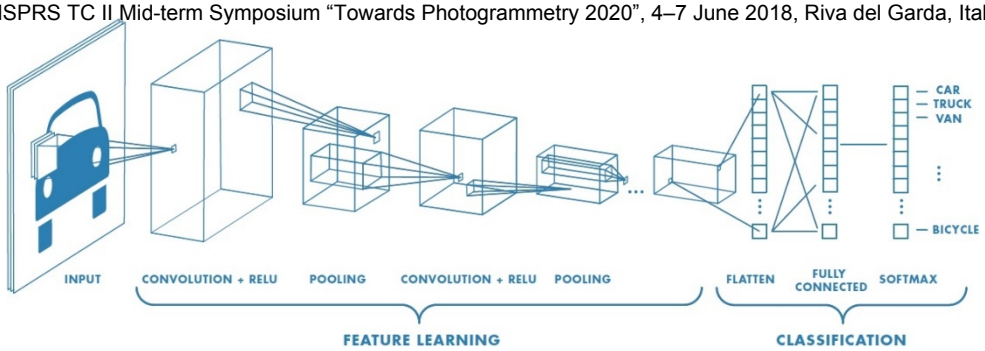

Figure 1: The architecture of a typical CNN. The first few layers usually consist of convolution and pooling layers. Units within the convolutional layers are then organised into feature maps where each unit is connected to local patches of the previous layer though a set of weights known as a filter bank. The result of this weighted sum is passed through a non-linearity (e.g. ReLU or Softmax function). Finally, a fully connected layer is added. The weights of the fully-connected layer are used to determine the classification of the inputted image. [Source: MathWorks., 2018]

In general, the application of CNNs can be split into three fundamental categories; classification, object detection and segmentation. Classification classifies the entire image into one category (e.g. dog, person, building etc.). Object detection is effectively thousands of classifications on one image. By doing this, the bounding box of the classification with the highest probability of a given object is assumed to be the objects location within an image. Lastly, typical segmentation algorithms (e.g. Long *et al*., 2015; Pinheiro *et al*., 2016; Badrinarayana *et al*., 2015) use a localised classification as the initial stage to perform pixel-wise semantic segmentation meaning every pixel within an image is given a class. In this paper, we discuss the potential of object detection. This decision is driven by the high performance of modern object detection algorithms. Whilst the reliability and robustness of semantic-segmentation is increasing within controlled benchmark datasets (e.g. PASCAL VOC (Everingham *et al*., 2010), Synthia (Ros *et al*., 2016), COCO (Lin *et al*., 2014)), there is much less evidence of applied general-purpose use to the same degree as object detection.

1.2 CNNs in civil in engineering:

Applied use of CNNs has been dominated by the computer science and medical imaging fields. Typical computer science applications include face detection (Parkhi *et al*., 2015; Li *et al*., 2015), automated driving/robotic vision systems (Huval *et al*., 2015) and general object detection used by scene categorisation for companies handling large image datasets (e.g. Facebook, Google, Amazon etc.). CNNs have also shown incredibly promising results within medical imaging with models now surpassing doctors for the classification of particular irregularities of cells (Zhang *et al*., 2017). However, CNNs still have a bounty of untapped potential within the engineering field.

Awareness of damage and defects in civil engineering structures is vital for appropriate prevention measures to be undertaken. However, due to the size of structures (e.g. roads, tunnels and tall buildings and bridges), regular surveying for defects is time consuming when carried out manually. Due to the inhomogeneity and complexity of defects such as cracks creating general-use detectors using traditional computer vision techniques (e.g. Abdel-Qader *et al*., 2003; Zou *et al*., 2012) has not been fully adopted by industry. Recent papers adopting deep learning CNN techniques however have shown highly accurate results for the detection and segmentation of cracks in a range of surfaces such as tarmac and asphalt (Zhang *et al*., 2016), metal fixings (Yeum and Dyke, 2015) and concrete (Cha *et al*., 2017; Koch and Brilakis, 2011). Here all papers reported crack identification with an accuracy of over 90% which greatly out performs prior attempts which do not utilise machine learning techniques. The results improve on papers such as Prasanna *et al*., (2016) where traditional machine learning algorithms such as Support Vector Machines and Random Forest were used for detection.

Similar to crack detection, railway track defect analysis has also benefitted from CNNs ability to detect inhomogeneous and variable features. Giben *et al*., (2015) demonstrated a method of mounting a camera to the front of a rail vehicle to capture imagery. Each image was subsequently segmented into each individual rail component with a 92% accuracy. This provided a promising foundation for further defect analysis. Faghih-Roohi *et al*., (2016) followed a similar approach, however, used video frames as input images for training and achieved similar accuracy (~91%). Soukup and Huber-Mörk, (2014) and Masci *et al*., (2012) also demonstrated similar approaches with high accuracy to detect defects in the steel tracks using a range of CNNs. These methods demonstrate the powerful nature of CNNs where consistent camera view-point positions are available.

Makantasis *et al*., (2015) demonstrated techniques to undertake a full tunnel defect inspection using CNNs. This demonstrates the ability of single model's ability to have a multi-use purpose through training on multiple classes. For example, the combination of defect inspection and Building Information Modelling (BIM) systems can be incorporated into a general solution, where both objects and defects are recorded simultaneously. Currently, BIM system generation procedures require a high level of manual feature labelling, this therefore presents a great potential for atomisation. While BIM tends to benefit from data acquired using terrestrial cameras, the ever-decreasing costs of Unmanned Aerial Systems (UASs) has also resulted in great potential for aerial imagery to be used for similar purposes. UASs are now becoming common-place on large engineering and surveying projects, and therefore offer a great platform for multi-use CNN applications. This can include tasks such as routine bridge and building inspection, power line surveillance, as well as feature mapping of built environments (Radovic *et al*., 2016). Other examples can include collection of building inventory data which is important for seismic risk assessment. Here, UASs offer potential for automating collection of important risk parameters such as the number of stories of a building (Liu *et al*., 2014). Furthermore, CNNs can offer the potential for automating the flight control of the UAS by allowing the system to perform object sighting and comprehension (object detection and classification respectively). This would be particularly effective when the feature of interest is moving, or its location is unknown prior to the flight (i.e. defects). Despite the potential for these solutions, engineering features can consist of very specialised equipment and therefore bypass the interest of the general public. To create such systems, the engineer or

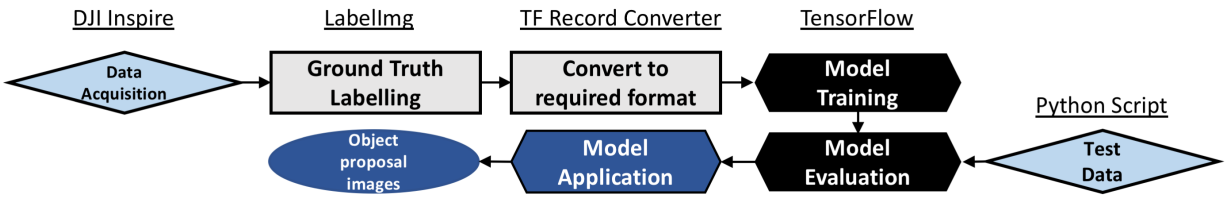

Figure 2: Workflow for training CNN with user acquired data. Test data is all the user data acquired that was not included in the training or validation datasets.

BIM professional would need to generate and train their own model fit for the purpose. A general consensus would indicate training a CNN requires two essential pre-requisites; a large dataset and high compute power. Whilst this consensus is true, it is possible for both to be overcome using a range of techniques described in this paper.

2. METHODOLOGY

The process of utilising a CNN for object detection can be categorised into the following stages; data acquisition, model training, model evaluation, model application (Figure 2). Here we describe the process taken to achieve each stage.

2.1 Data Acquisition

The aim of the experiment was to create a site-specific object detection model. Therefore, data was collected of the site. Our data was acquired using a DJI Zenmuse X5 camera mounted onto a DJI Inspire 1 UAS. As the original purpose of the data collection was for a photogrammetric reconstruction, both nadir and oblique imagery was taken to aid an accurate camera calibration. Both nadir and oblique images were used for training, validation and testing of the model. To further increase the robustness and versatility of the training data, 125 images of railway tracks in varying light conditions and view angles were downloaded from the internet and included into the training set. LabelImg (Tzutalin, 2015) was used to create ground-truth bounding boxes. As the railway track is continuous and has an inconsistent shape dependant on intersections etc. it was decided to define a section of rail as a ~3m2 section containing the steel tracks and concrete/wooden sleepers. 150 images of the railway track survey were used for training making the complete training set 275 images. To further utilise the small set of images, data augmentation was used. Data augmentation is a common technique in CNNs to fully-utilise small and large datasets, as well as avoid potential over-fitting of the model. Here, we applied a horizontal and vertical flip as well as a contrast adjustment to the input images. This subsequently tripled the image dataset. The total process of generating ground-truth labels took ~2 hours.

2.2 Network Architectures and training

Google's TensorFlow (Abadi., 2016) was used as the framework for model training. All training was undertaken on a single Nvidia 1080Ti Graphical Processing Unit (GPU). Object detection architectures can be categorised into two main categories; two-stage and one-stage. Two-stage approaches were popularised by the R-CNN architecture (Girshick *et al*., 2013). The algorithm works by first generating a regional proposal for potential bounding box locations, the second stage then classifies each proposal candidate using a typical CNN classification. Lastly, a refinement stage is undertaken to eliminate duplicate bounding boxes and the proposals with the highest classification probability is used as the object location. This method has since been refined (Girshick *et al*., 2014; Ren *et al*., 2015; He *et al*., 2017) and has consistently achieved state-of-the-art results on the COCO benchmark challenge. However, one of the biggest limitation with two-stage architectures is their inability to optimise or parallelise. Although accurate, currently real-time object-detection using a two-stage approach is not feasible (Lin *et al*., 2017). One-stage detectors speed up this process by applying a regular dense sample of classifications over the image at various scales and aspect ratios. Each sample consists of a bounding box for which the highest classification scores are recorded. This not only makes the process computationally cheaper, but also allows for parallelisation over multiple cores. The most common one-stage detectors consist of the YOLO (Redmon *et al*., 2016; Redmon and Farhadi, 2016) and SSD (Liu *et al*., 2016) detectors. Despite significant increases in detection speed, the YOLO and SSD accuracies typically fall within 10-40% of state-of-the-art two-stage detectors (Huang *et al*., 2017; Lin *et al*., 2017). Here, we compare the recent RetinaNet (Lin *et al*., 2017) one-stage object-detector which claims similar accuracies to state-of-the-art two-stages detectors using its novel focal loss algorithm. For comparison, the commonly used Faster RCNN with ResNet (Chen and Gupta, 2017) and Inception (Szefedy *et al*., 2017) were also trained with the dataset.

2.2.1 Faster RCNN

Faster RCNN works by first creating a region proposal network. This is achieved processing the images with a features extractor (in our case ResNet and Inception-v4) at an intermediate level (e.g. ~Conv4-Conv6 layer). The results of these predictions attempt to classify the image into class-agnositc box proposals of foreground and background. The second stage takes the box proposals and uses them to crop from the same intermediate feature map. The remainder of the feature extractor is passed to the proposal to predict class and class-specific box refinement (Girshick, 2015). The location loss function used is the SmoothL_1 for training where:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{classes}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (1)$$

Where:

$\{p_i\}$ is the predicted probability of proposal being an object, $\{t_i\}$ is the coordinates of the predicted bounding box, L_{cls} is the log loss, p_i^* is the ground truth objectness label, L_{reg} is the SmoothL_1 loss and t_i^*, is the true box coordinates. The loss function of the regressor can then be defined as:

$$\text{SmoothL}_1(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x|-0.5 & \text{otherwise,} \end{cases} \quad (2)$$

Whilst we separately use the ResNet (He *et al*., 2016) and the Inception-v4 with ResNet (Szegedy *et al*,. 2016) CNN classifiers, an in-depth description of these is beyond the scope of this paper.

2.2.2 Focal Loss for Dense Object Detection (RetinaNet)

RetinaNet was published along with the novel algorithmic concept of focal loss (Lin *et al*., 2017). Lin *et al*. (2017) propose that a large class imbalance between foreground and background (i.e. 1:1000) during the training of dense object detectors negatively impact accuracy. They address this issue by re-shaping the cross-entropy loss such that it down weights the loss assigned to well-classified features during training. Here, the problematic well-classified features refer mostly to easily identified negatives (background). The reshaped focal loss differentiates between easy and hard examples and then uses this differentiation to down weight easy examples, focussing training on the hard examples. The focal loss starts from a Cross Entropy (CE) binary classification:

(3)

$$\mathrm{CE}(p,y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise,} \end{cases}$$

Which can be rewritten as:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ (1-p) & \text{otherwise,} \end{cases}$$

(4)

Where $\mathrm{CE}(p,y) = \mathrm{CE}(p_t) = -\log(p_t)$

The weighting is then applied by adding a modulating factor $(1-p_t)^{\gamma}$ with tunable focussing parameter $\gamma \geq 0$. The focal loss function is then defined as:

(5)

$$(p_t) = -(1-p_t)^{\gamma}\log(p_t)$$

This improvement allowed for an Average Precision (AP) score of 32.5 with speed 73ms on the COCO dataset. In comparison, Faster RCNN with ResNet and Inception-ResNet achieved an AP scores of 33.3 (speed 106ms) and 37 (speed 620ms) respectively. The RetinaNet classification is carried out using a Feature Pyramid Network (Lin *et al*., 2017) backbone on top of a feedforward ResNet architecture.

2.3 Model hyper parameters

Hyper parameters need to be set for any CNN the most influential and important here are; batch size, learning rate, decay, epochs and dropout. Simply speaking, batch size determines the number of images passed through one single forward/backward pass of the network. Learning rate is a constant that determines how quickly the network can abandon what it has learnt prior for new information. Decay is the ratio between learning rate and epoch, where, as the epoch increases the learning rate decreases. This is calculated by applying the following:

(6)

$$\text{decayed learning rate} = \text{learning rate} * \text{decay rate}^{\left(\frac{\text{global step}}{\text{decay step}}\right)}.$$

An epoch refers to a single presentation of all training data through the network. Dropout is a regularisation technique to avoid overfitting of a model. This was first outlined by Hinton *et al*., (2012) who showed by randomly omitting half of the feature detectors on each training case result degradation caused by overfitting could be mitigated. Hyper parameters can be tuned intuitively depending on the model response to training data. The hyper parameters used for each network architecture are shown in Table 1.

2. 4 Model performance evaluation

Initially, to quantify accuracy of the trained models the mean (mAP) was computed with a true positive being defined as the proposal and ground-truth having an Intersection Over Union (IOU) greater than 0.5. However, as the labelling of ground truth was computed as ~3m2 extracts of railway track this caused a significant under-estimation of the accuracy. For example, a section of railway track may be completely covered in object proposals, however, if these were not the initial ground truth labels (which would be difficult as square sections of railway track are largely homogeneous), the metric may define this as inaccurate. We addressed this issue with two approaches. Firstly, a second evaluation was performed on the model at each epoch with an IoU threshold of 0.1. Whilst this largely solves the issue described above, it introduces an uncertainty with respect to poorly detected areas perpendicular to the track (figure 3b). Despite this, however, we argue this is a more representable accuracy metric of the results discussed later. Finally, we derive a third metric which does not make use of the IoU. Here, the amount of track covered by object proposals was compared as a ratio to the level of track not covered by an object proposal. We computed this by measuring the length of track within and not within object proposals. This also enabled an accuracy assessment to be undertaken on test images as well as validation images. A total of 500 test images were used for evaluation which were captures during the same aerial survey as the training data.

Figure 3: Issues relating to Intersection over Union (IoU) accuracy measurement. A) Unfortunate positioning of ground truth boxes results in the image having an accuracy of 50%@IoU=0.5. In the image a total of 78% of the railway track is covered by prediction boxes. With an IoU=0.1 the accuracy is increased to 75% B) Limitations of IoU@0.1 metric. Although ~10% of railway track is detected the accuracy measurement would be 100%@IoU=0.1. C) Demonstration of more representative accuracy measurements. Image would have 0%@IoU=0.5 and 100%@IoU=0.1.

Table 1: Hyper parameters for model training. Dropout rate is a fraction proportion of the feature detectors. Training time is on a single Nvidia 1080Ti GPU.

Network	Batch Size	Learning Rate	Decay	Dropout Rate	Epochs	Training Time (hrs)
Faster RCNN with ResNet	4	3E-4	0.96	0.5	200	4
Faster RCNN with Inception-v4	4	3E-4	0.96	0.5	200	5.5
RetinaNet	4	1E-5	0	0.5	200	6.5

3. RESULTS

Initial training of the CNN (figure 4) demonstrated that whilst the model had appeared to converge with a small loss (~0.4), accuracy@IoU=0.5 was relatively low to what is expected in comparison to the cited studies of the network architectures. This we feel is a practical demonstration of the unsuitability of the accuracy metric used (IoU=0.5). Upon review it was noted that the railway track was detected in every training/validation and test image, further suggesting the unsuitability of the accuracy metric. Furthermore, accuracy is noticeably higher in the first few epochs and then decreases steeply before apparently settling. This is a common sign of a model ‘over-fitting’ to the training data (Wu and Gu, 2015). We discuss this further later on.

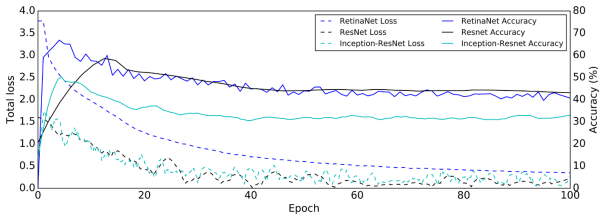

Figure 4: Training loss and validation accuracy (IoU=0.5) for each of the network architectures. Accuracy is comparably lower to papers cited in the methodology section. The training loss appears to have converged well.

The model validation was re-evaluated for each epoch with the accuracy metric IoU=0.1 (figure 5). A significant improvement is demonstrated as suggested. The accuracy is on average ~2.25x higher than the corresponding accuracy measurement where IoU=0.5 for each network. Furthermore, a visual review of the results (figure 7) showed no evidence of incorrect railway track predictions (where the side of the track is predicted (figure 3b)) being evaluated as correct.

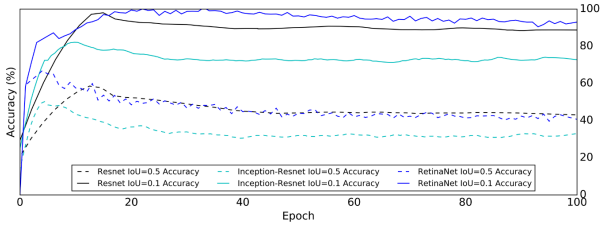

Figure 5: IoU comparison where IoU=0.5 and 0.1 thresholds are set for evaluation of the model performance. Results are approximately ~2.25x greater when IoU=0.1.

To assess if the model was over-fitting, the training accuracy was evaluated for each epoch (figure 6). The training evaluation is calculated by running a single evaluation of each image in the training dataset at the end of each epoch. The evidence of over-fitting is further demonstrated here where the decline in validation accuracy is mirrored by an increase in training accuracy. Whilst for a general fit model this would be a very worrying sign, for a site-specific rapid object detection system where little variation between training and testing data is likely to exist we argue this is not a major issue.

The final evaluation method we carried out was a percentage ratio of railway track covered by object-detection proposals and railway track uncovered. The results along with the IoU results are displayed in table 2. Each architecture achieved a high coverage (>75%) of the railway track. Oblique images in general performed worse, most likely due to the more limited quantity of training data. Overall, the one-stage Retinanet outperformed the two-stage feature extractors (Resnet and Inception-Resnet). This was evident in both the IoU=0.1 and percentage proposal coverage evaluation methods. At IoU=0.5 the Resnet model was the highest performer. As one would expect, the one-stage Retinanet was also significantly quicker at creating proposals. Proposals were computed on a Nvidia 1060 GPU. In this scenario, Retinanet was 11.7x faster than Inception-Resnet and 5.8x faster than Resnet for computing proposals. Inception-Resnet was the slowest performing architecture and also yields in the poorest results in each evaluation.

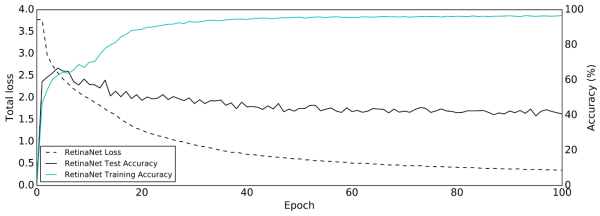

Figure 6: Training and Validation evaluation accuracies at IoU=0.5. The simultaneous increase in training accuracy and decrease in validation accuracy is a classic sign of over-fitting of the training data.

4. DISCUSSION

The results demonstrate the potential for training high-accuracy, site-specific object detection models in a short period of time. This was achieved by utilising a small sample (150) of images acquired using a UAS as training data. The results obtained are only made possible by the ability to bottleneck (or fine-tune) the model architecture. Therefore, each model was not trained from scratch (i.e. all weights initialised randomly). Instead, the publicly available weights from training on the ImageNet dataset were used as initial parameter weights. Without this, initial training would require a significantly larger dataset (i.e. > 20,000 images) and a much longer training time (i.e. ~1-2 weeks). However, weight information is readily available for the majority of common architectures and here we demonstrate it can be ‘re-purposed’ for an engineering scenario. Furthermore, we demonstrate that with the aid of googles TensorFlow and labelImg, a training dataset has to potential to be labelled and trained within a day.

Table 2: Results of each accuracy metric for network architectures. Computed proposal times on a Nvidia 1060 GPU are also shown. Retinanet can be seen here to be the highest performing network architecture.

	Resnet	Inception-Resnet	Retinanet
Proposal Coverage (%)	83.1	75.6	91.3
Accuracy@IoU=0.5 (%)	43.2	32.9	40.7
Accuracy@IoU=0.1 (%)	87.9	72.4	93.0
Proposal Time (seconds)	0.64	1.29	0.11

The accuracy of the models demonstrated a high level of reliability in the solution. Despite initial validation accuracy (@IoU=0.5) of 43.2%, 32.9% and 40.7% for Resnet, Inception-Resnet and Retinanet respectively, these low scores were most certainly due to the unsuitability of the accuracy metric. This was quantifiably backed up by increased accuracies in the two other accuracy metrics. The first of which decreased the IoU threshold value from 0.5 to 0.1. This new threshold resulted in evaluation accuracies of 87.9%, 72.4% and 93% for Resnet, Inception-Resnet and Retinanet respectively. However, this does open up the potential risk for poorly positioned predictions that sit on the side of the railway track but near it being classed as correct predictions. However, a visual inspection of the results showed no evidence of this occurring. This is most likely due to the unique structure of the track being a prime subject for training a CNN. This is also likely to be the main reason high accuracies were obtained for the percentage of track covered accuracy metric. Here, we established a new accuracy assessment as the tracks demonstrated a potential problem for the IoU method. This was caused by railway tracks being a continuous feature and therefore object labelling consisting of ~3m2 segments of track. As this was done randomly, there was no way for the CNN to determine where the original ground-truth box would have lay. Although we feel this justifies reducing the IoU threshold to 0.1, it is perhaps still not the most relevent method. By measuring the length of the track and dividing it by the length of track covered by object proposals we develop an accuracy metric which determines how much of the track has been successfully identified. Interestingly, the one-stage object detection Retinanet performed the best overall. The results obtained were 83.1%, 75.6% and 91.3% for Resnet, Inception-Resnet and Retinanet respectively. The superior proposals are also computed in a significantly shorter time (0.11 seconds).

The ability for high accuracies with very low compute times opens up a large potential for reliable real-time object detection. With the decreasing costs of UAS's along with ever improving libraries and API's for flight control, the incorporation of real-time object detection for intelligent flight control is clearly achievable. Such flight control could be used for automatic intelligent navigation solutions for both continuous structures such as roads and tracks as well as moving objects such as vehicles. We further argue that this is not only achievable, but object-detection models can be trained and deployed quick enough that it is a reasonable solution for many applications (e.g. defect detection, BIM inventory assessments, safety inspections etc.).

The work here demonstrates a typical object-detection system. However, for certain tasks a full semantic segmentation is sought after. The initial object-detection can act as a mask to further perform pixel-wise segmentation and classification through deep learning methods (e.g. Long *et al*., 2014; Pinheiro *et al*., 2016). Whilst the field of semantic segmentation through deep learning is advancing rapidly, there has not been any real demonstration of robust systems which could be incorporated easily into engineering workflows. This is mainly due to the increased complexity of the problem along with the increased labour of accurately labelling data. Object detection does still offer exciting possibilities for pixel-wise semantic segmentation using traditional image processing techniques. Many traditional image processing techniques often fail when unexpected background in introduced into the problem (e.g. covered objects, unexpected surface types etc.). By detecting object boundaries this problem can be simplified by an order of magnitude.

The results indicate that the Inception-Resnet architecture performed the worst in respect to both speed and accuracy. This demonstrates that the speed-accuracy trade-off is not always a relevant assessment of a model's potential. The performance of the one-stage Retinanet perhaps indicates that the incorporation of an updated cross-entropy function to include a focal loss strategy is highly effective. The novelty of the algorithm should however still suggest caution should be taken when reaching conclusions here. Neural networks are highly complicated systems and still highly unpredictable. The performance of one dataset is by no means a guarantee of the same model's performance on a different dataset. Such caution is particularly relevant when generating many models based on single-class systems which vary from the ImageNet classes.

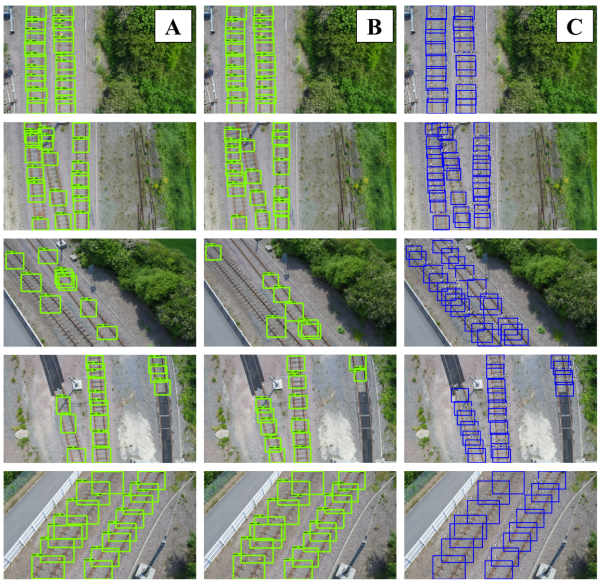

Figure 7: Visual inspection of computed proposals for **A)** Inception-Resnet, **B)** Resnet and **C)** Retinanet. In general, oblique imagery performed worse than directly nadir imagery. This is likely due to a larger number nadir imagery in the training dataset.

4.1 Future work

Another interesting application would utilise the initial predictions computed through the object detection model to map continuous features such as railway track into GIS. This could be achieved by fitting lines between proposals to determine the direction and the features centre point. By then further combining the predicted features and the on-board GNSS a potential very accurate vector file of the feature could be produced. Such developments could be very useful for developing countries that do not benefit from well managed GIS.

5. CONCLUSION

The work presented here demonstrates the potential application of rapid development of CNNs for single-class object detection systems. Each CNN was trained using 275 images of railway track acquired from a UAS and online-resources. Whilst the work is directed at single-class (railway track) there is no reason to believe the solution is not scalable and with more training data, more classes could be trained. The Faster RCNN two-stage object detection was tested with two network architectures for

classification (Resnet and Inception-Resnet). A third one-stage object detection which incorporates a focal loss mechanism, using the novel Retinanet architecture was also evaluated. Retinanet out-performed both two-stage object detectors in respect to both speed and accuracy. Three accuracy metrics were computed to evaluate model performance which included; IoU@0.5, IoU@0.1, track proposal coverage. The latter refers to a percentage of railway track covered by object detection proposals compared to track length. Retinanet achieved an accuracy of 91.3% track coverage, with Resnet and Inception-Resnet achieving 83.1% and 75.6% respectively. Manual ground-truth labelling took ~2 hours and model training between four and six hours on a single Nvidia 1080 Ti GPU. The models made use of fine-tuning pre-trained weights to achieve high accuracies in a short period of time. The results show to potential for such rapid object detection systems, which have a wide-range of potential applications that can the engineering field could benefit. Further research could develop the processes discussed within this paper for a range of applications such as; pixel-wise segmentation, automatic intelligent UAS navigation systems and continuous feature mapping systems.

ACKNOWLEDGEMENTS

The authors would like to thank Bentley Systems for supplying the UAS imagery used for model training and evaluation.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. TensorFlow: A System for Large-Scale Machine Learning. In OSDI (Vol. 16, pp. 265-283).

Abdel-Qader, I., Abudayyeh, O. and Kelly, M.E., 2003. Analysis of edge-detection techniques for crack identification in bridges. Journal of Computing in Civil Engineering, 17(4), pp.255-263.

Badrinarayanan, V., Kendall, A. and Cipolla, R., 2015. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561.

Chen, X. and Gupta, A., 2017. An implementation of faster rcnn with study for region sampling. arXiv preprint arXiv:1702.02138.

Cha, Y.J., Choi, W. and Büyüköztürk, O., 2017. Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. Computer-Aided Civil and Infrastructure Engineering, 32(5), pp.361-378.

Chen, L., Wang, S., Fan, W., Sun, J. and Naoi, S., 2015, November. Beyond human recognition: A CNN-based framework for handwritten character recognition. In Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on (pp. 695-699). IEEE.

Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L., 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE.

Everingham, M., Van Gool, L., Williams, C.K., Winn, J. and Zisserman, A., 2010. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), pp.303-338.

Giben, X., Patel, V.M. and Chellappa, R., 2015, September. Material classification and semantic segmentation of railway track images with deep convolutional neural networks. In Image Processing (ICIP), 2015 IEEE International Conference on (pp. 621-625). IEEE.

Girshick, R., 2015. Fast r-cnn. arXiv preprint arXiv:1504.08083.

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017, October. Mask r-cnn. In Computer Vision (ICCV), 2017 IEEE International Conference on(pp. 2980-2988). IEEE.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K., 2017, July. Speed/accuracy trade-offs for modern convolutional object detectors. In IEEE CVPR.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R. and Mujica, F., 2015. An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716.

Koch, C. and Brilakis, I., 2011. Pothole detection in asphalt pavement images. Advanced Engineering Informatics, 25(3), pp.507-515.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), pp.541-551.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E. and Jackel, L.D., 1990. Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems (pp. 396-404).

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. Nature, 521(7553), pp.436-444.

Li, H., Lin, Z., Shen, X., Brandt, J. and Hua, G., 2015. A convolutional neural network cascade for face detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5325-5334).

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In CVPR (Vol. 1, No. 2, p. 4).

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. arXiv preprint arXiv:1708.02002.

Liu, P., Chen, A.Y., Huang, Y.N., Han, J.Y., Lai, J.S., Kang, S.C., Wu, T.H., Wen, M.C. and Tsai, M.H., 2014. A review of rotorcraft unmanned aerial vehicle (UAV) developments and applications in civil engineering. Smart Structures and Systems, 13(6), pp.1065-1094.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

Long, J., Shelhamer, E. and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Makantasis, K., Protopapadakis, E., Doulamis, A., Doulamis, N. and Loupos, C., 2015, September. Deep convolutional neural networks for efficient vision based tunnel inspection. In Intelligent Computer Communication and Processing (ICCP), 2015 IEEE International Conference on (pp. 335-342). IEEE.

Masci, J., Meier, U., Ciresan, D., Schmidhuber, J. and Fricout, G., 2012, June. Steel defect classification with max-pooling convolutional neural networks. In Neural Networks (IJCNN), The 2012 International Joint Conference on (pp. 1-6). IEEE.

Parkhi, O.M., Vedaldi, A. and Zisserman, A., 2015, September. Deep Face Recognition. In BMVC (Vol. 1, No. 3, p. 6).

MathWorks., 2018. Convolutional Neural Networks. Available Online: https://uk.mathworks.com/discovery/convolutional-neural-network.html

Pinheiro, P.O., Lin, T.Y., Collobert, R. and Dollár, P., 2016, October. Learning to refine object segments. In European Conference on Computer Vision (pp. 75-91). Springer, Cham.

Prasanna, P., Dana, K.J., Gucunski, N., Basily, B.B., La, H.M., Lim, R.S. and Parvardeh, H., 2016. Automated crack detection on concrete bridges. IEEE Transactions on automation science and engineering, 13(2), pp.591-599.

Radovic, M., Adarkwa, O. and Wang, Q., 2017. Object Recognition in Aerial Images Using Convolutional Neural Networks. Journal of Imaging, 3(2), p.21.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Redmon, J. and Farhadi, A., 2016. YOLO9000: better, faster, stronger. arXiv preprint, 1612.

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

Ros, G., Sellart, L., Materzynska, J., Vazquez, D. and Lopez, A.M., 2016. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3234-3243).

Soukup, D. and Huber-Mörk, R., 2014, December. Convolutional neural networks for steel surface defect detection from photometric stereo images. In International Symposium on Visual Computing (pp. 668-677). Springer, Cham.

Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A., 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In AAAI (pp. 4278-4284).

Tzutalin. LabelImg. Git code (2015). Available Online: https://github.com/tzutalin/labelImg

Wu, H. and Gu, X., 2015. Towards dropout training for convolutional neural networks. Neural Networks, 71, pp.1-10.

Yeum, C.M. and Dyke, S.J., 2015. Vision-based automated crack detection for bridge inspection. Computer-Aided Civil and Infrastructure Engineering, 30(10), pp.759-770.

Zhang, Z., Xing, F., Su, H., Shi, X. and Yang, L., 2017. Recent Advances in the Applications of Convolutional Neural Networks to Medical Image Contour Detection. arXiv preprint arXiv:1708.07281.

Zoph, B. and Le, Q.V., 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

Zou, Q., Cao, Y., Li, Q., Mao, Q. and Wang, S., 2012. CrackTree: Automatic crack detection from pavement images. Pattern Recognition Letters, 33(3), pp.227-238.