Preprint submitted to the
Sixth International Congress on Information and Communication Technology (ICICT 2021), London, UK,
to be published in Advances in Intelligent Systems and Computing (Springer Nature).

FasteNet: A Fast Railway Fastener Detector

Jun Jet Tai[1], Mauro S. Innocente[1], and Owais Mehmood[2]

[1] Coventry University, Priory St, Coventry CV1 5FB, UK
taij@uni.coventry.ac.uk,
Mauro.S.Innocente@coventry.ac.uk, https://availab.org/
[2] Omnicom Balfour Beatty, Clifton Park Ave, Rawcliffe, York YO30 5PB, UK
owais.mehmood@balfourbeatty.com

Abstract. In this work, a novel high-speed railway fastener detector is introduced. This fully convolutional network, dubbed FasteNet, foregoes the notion of bounding boxes and performs detection directly on a predicted saliency map. Fastenet uses transposed convolutions and skip connections, the effective receptive field of the network is 1.5× larger than the average size of a fastener, enabling the network to make predictions with high confidence, without sacrificing output resolution. In addition, due to the saliency map approach, the network is able to vote for the presence of a fastener up to 30 times per fastener, boosting prediction accuracy. Fastenet is capable of running at 110 FPS on an Nvidia GTX 1080, while taking in inputs of 1600×512 with an average of 14 fasteners per image. Our source is open here: https://github.com/jjshoots/DL_FasteNet.git

Keywords: Object Detection, Convolutional Neural Networks, Railway Fastener Detection

1 Introduction

Traditionally, the inspection of railway fasteners has been done by visual inspection [1]. However, modern inspection methods are taking over this task. Examples include the Viola-Jones algorithm [2] and symmetry-based pyramid histogram of oriented gradients (PHOG) algorithm [3]. Gibert et. al. used support vector machines (SVMs) to perform fastener and defect detection [4]. The work was also extended to fully convolutional neural networks [5]. Comparisons show that deep learning methods work well even with limited training data while being easier to design. More crucially, deep learning models tend to have lower inference time and good generalizability. Recent work by Song et. al. [6] utilized a ResNet18 [7] backbone network on top of Faster-RCNN [8]. Their implementation achieved a recall and precision of more than 99% but was capped at 35 FPS on an Nvidia Titan X for 210 fasteners detected per second. Wang et. al. proposed using the popular YOLOv2 and YOLOv3 architectures to perform fastener component detection [9]. Their implementation on an Nvidia GTX 1080Ti worked on close up images of railway fasteners, and performed with a mean average precision (mAP) of 93% at 35 FPS. These works show that deep convolutional neural networks (DCNNs) can be applied to railway fastener detection, like other computer vision tasks before it. However, the task of fastener, defect and component detection is unique. Generally, the scale and rotation of the fasteners do not change, there is also no overlap between fasteners. This is true even in the case of sleeper detection. This is in contrast to general object detection tasks that may be subject to scale variance, occlusion, overlap, and other visual interruptions. For this reason, many existing network architectures are inelegant and can be improved. This work introduces several novel ideas:

- Bounding boxes prevent the network from making more than one prediction per fastener without output suppression. Even so, predicting more than 10 bounding boxes at every output spatial location is excessive. FasteNet instead predicts a saliency map. The bounding boxes are then derived from this output using a first principles method, boosting accuracy by allowing the network to predict up to 30 times per fastener without requiring more than a one channel output feature map.
- There is no need to train the network to work on a large range of fastener scales. Due to this, FasteNet has an effective receptive field (ERF) only around 50% larger than necessary.
- FasteNet has different regimes during training and inference. During inference, outputs are thresholded and ceiled to increase prediction accuracy by eliminating low confidence predictions. During training, this step is removed to allow the network to fully train for every output location.

The rest of this paper is arranged as follows; in Section 2, our proposed system architecture is presented. In Section 3, the novelties that improve system performance are introduced. Finally, the paper is concluded in Section 4.

2 Methodology

2.1 Problem Description

FasteNet's purpose is to count fasteners in an image; not novel, but serves as a proof of concept for the ideas introduced here. A set of 997 images are generated by a New Measurement Train, some examples are shown in Fig. 1. The fasteners in the images include fasteners which are occluded, damaged or missing. Furthermore, the fasteners are exposed under various lighting conditions due to time of day; no artificial illumination is used. The images also contain fasteners of various types - PR clip, E clip, Fast clip, C clip, and J clip. The images are solely black and white, making detection harder by removing colour information.

The problem is treated as a localization one. As long as the network prediction area overlaps with a ground truth bounding box, the prediction is considered a true positive (TP), otherwise it is perceived as a false positive (FP). The network must output the number of fasteners it sees in the image. This is to enforce the idea that the algorithm must clearly distinguish one fastener from another fastener, as opposed to simply outputting a saliency map. This variation of object detection strays away from the general bounding box detection one [10] [11] [8]. As long as the the prediction area lies over a real fastener is sufficient.

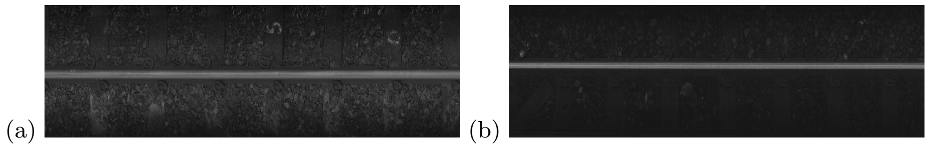


Fig. 1. Example of railway images in the dataset where (a) all fasteners are visible, (b) the image is very badly illuminated.

2.2 FasteNet Design

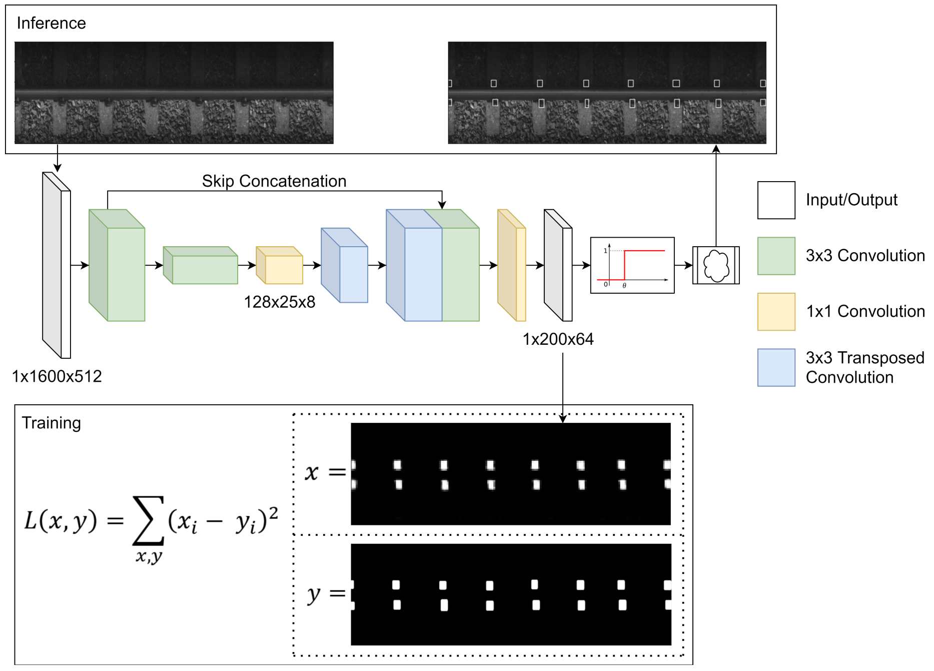


Fig. 2. Top level view of FasteNet. During training, the network trains against segmentation masks. During inference, the network outputs are thresholded before a contour finding algorithm is applied to find individual fasteners.

Fig. 2 provides a top level overview of FasteNet's architecture, as well as training and inference regimes. The input to the network is a black-and-white image defined here as $w_i \in \mathbb{R}^{1\times \mathrm{m}\times \mathrm{n}}$ where $m \leftrightarrow 64|m$ and $n \leftrightarrow 64|n$. It outputs a saliency map of $x_i \in \mathbb{R}^{1\times \frac{\mathrm{m}}{8}\times \frac{\mathrm{n}}{8}}$. The backbone network of FasteNet has just-enough layers to provide it with a roughly 64×64 ERF. This is a slightly hand-wavy method of calculating the ERF; in reality, the ERF more closely represents a Gaussian distribution on the true receptive field [12]. This ERF is roughly 1.5×

the size of a fastener in the image. However, using only convolutional and pooling layers to achieve this ERF would translate to a very poor output resolution of 1/64th the input resolution. Many implementations of object detection work well with low output resolutions by using multiple bounding boxes per spatial element, and then suppressing repeat predictions [13] [10]. This approach relies on very deep networks to provide the the output layers with enough semantic value, and then uses bounding boxes to handle localization precision loss.

Our approach takes a different look at the problem. FasteNet uses transposed convolutions to increase the output spatial resolution to 200×64 (1/8th the input size), and then predicts an objectness score at each output location. To reduce information loss and alleviate the vanishing gradient problem [14], feature maps from an earlier layer are concatenated to layers before the output layer. 1×1 convolutions are then used to produce the output saliency map. This allows up to 30 predictions per fastener. Other tidbits that boost FasteNet's performance include batch normalization to combat internal covariate shift during training and to provide regularization. Leaky ReLU units with a negative gradient of -0.1 are used to prevent dead neurons. The overall network architecture is shown in Table 1, note that the spatial size depends on the input size; any input size divisible by 64 can be computed by FasteNet - our implementation uses an input size of 1600×512 *during inference.*

During training, FasteNet trains against semantic segmentation masks. An example is shown in Fig. 2. During inference, FasteNet thresholds all output pixels, $x_i \leftarrow \delta_{[x>\theta]}$, where δ is the Kronecker delta, and then uses a boundary walking algorithm by Suzuki et. al. [15] to identify each cluster at the output. The cluster bounds are used to derive bounding boxes. Semantically, given a chain of points $c_i \in \mathbb{R}^{\mathrm{n} \times 2}$ denoting the contour for a cluster, the top left $\mathbf{b}_1$ and bottom right $\mathbf{b}_2$ corners of the bounding boxes are defined as $\mathbf{b}_1 = \{\min_{i \in n} \mathbf{c}_{i_x}, \max_{j \in n} \mathbf{c}_{j_y}\}$ and $\mathbf{b}_2 = \{\max_{i \in n} \mathbf{c}_{i_x}, \min_{j \in n} \mathbf{c}_{j_y}\}$. Other algorithms for drawing bounding boxes can also be used. In an extreme case, bounding boxes can be foregone exclusively, and the shape of the contour itself can be used to identify instances of fasteners.

2.3 Training and Evaluation

The network is trained with 1,000 256×256 random crops per image for 700 out of 997 available images. The mean squared error (MSE) objective is used, $L(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{x},\mathbf{y}} (x_i - y_i)^2$, where $\mathbf{x}$ and $\mathbf{y}$ represent the network prediction and ground truth mask respectively. The Adam optimizer [16] with a learning rate of 1e-6 and weight decay of 1e-2 is set as the optimizer. The network converges after the first 20 epochs with a precision of 96% and a recall of 89% when validated on the remaining 297 images. Hard negative mining was then used to train the network on difficult training images (more than 1 FP or FN). The network was then explicitly trained on these examples for 2 epochs, before continuing on the whole 700 images for another 2 epochs. This was repeated 3 times. In the end, a precision of 99% and a recall of 90% was accomplished on the validation dataset.

Table 1. Network architecture of FasteNet. The outputs from layer 3 are concatenated channel wise to outputs of layer 10 to form the input to layer 11.

Layer #	Operation	Input Size	Output Size	Kernel Size	Pad Size	Pool Size	Activation
1	Convolution	1@ 1600×512	32@ 800×256	3×3	1×1	2×2	Leaky ReLU
2	Convolution	32@ 800×256	32@ 400×128	3×3	1×1	2×2	Leaky ReLU
3	Convolution	32@ 400×128	64@ 200×64	3×3	1×1	2×2	Leaky ReLU
4	Convolution	64@ 200×64	64@ 100×32	3×3	1×1	2×2	Leaky ReLU
5	Convolution	64@ 100×32	128@ 50×16	3×3	1×1	2×2	Leaky ReLU
6	Convolution	128@ 50×16	128@ 25×8	3×3	1×1	2×2	Leaky ReLU
7	Convolution	128@ 25×8	64@ 25×8	1×1	-	-	Leaky ReLU
8	Transposed convolution	64@ 25×8	64@ 50×16	4×4	1×1	-	Leaky ReLU
9	Transposed convolution	64@ 50×16	64@ 100×32	4×4	1×1	-	Leaky ReLU
10	Transposed convolution	64@ 100×32	64@ 200×64	4×4	1×1	-	Leaky ReLU
11	Convolution	128@ 200×64	1@ 200×64	1×1	-	-	Sigmoid

3 Performance Analysis

The model is validated on the remaining 297 images from the dataset. Evaluation with metrics done by other authors [5] [4] [17] [18] is not possible as the datasets are not public and the detection classes are different. Instead, FasteNet is compared to two other internal models. This is to validate our hypothesis that a network designed in this manner has better performance than a brute force DCNN implementation. These two networks are denoted here as VanillaNet and LargeNet, shown in Table 3 and Table 4 with a comparison in Table 2.

VanillaNet does not utilize transposed convolutions and consists of 11 layers. LargeNet uses transposed convolutions with a larger ERF of roughly 128×128. Note that because of the output scaling ratio of this network, the inputs are cropped to 1536×512. These two additional networks are trained on the same training regime that was used on FasteNet in Section 2.3. Special considerations are made to ensure that each network has roughly the same number of parameters to reduce variance in representational capacity.

By varying the threshold θ, the rate of predictions can be varied to form the precision recall curves for each network shown here in Fig. 3. The performance of FasteNet is on top. LargeNet and VanillaNet implementations may perform as well or better than FasteNet when given more training time due to deeper network architectures and hence theoretically more representational capacity. However, this sub-percentage performance increase is compensated by the loss in some computational speed, which is especially the case in VanillaNet. At the end of training, FasteNet is able to output bounding boxes around each fastener as shown in Fig. 4.

4 Conclusion

In this work, a methodology for designing DCNN models for railway maintenance tasks was introduced. Formally, bounding box regression is replaced with

Table 2. Summary of FasteNet in comparison with two other networks built for an evaluation of FasteNet's design.

Network	Parameter Count	FLOPS	FPS
FasteNet	494k	4.00G	110
VanillaNet	500k	25.93G	25
LargeNet	514k	6.38G	80

Table 3. Network architecture of VanillaNet.

Layer #	Operation	Input Size	Output Size	Kernel Size	Pad Size	Pool Size	Activation
1	Convolution	1@1600×512	32@800×256	3×3	1×1	2×2	Leaky ReLU
2	Convolution	32@800×256	64@800×256	3×3	1×1	-	Leaky ReLU
3	Convolution	64@800×256	32@800×256	3×3	1×1	-	Leaky ReLU
4	Convolution	32@800×256	64@400×128	3×3	1×1	2×2	Leaky ReLU
5	Convolution	64@400×128	128@400×128	3×3	1×1	-	Leaky ReLU
6	Convolution	128@400×128	64@400×128	3×3	1×1	-	Leaky ReLU
7	Convolution	64@400×128	128@200×64	3×3	1×1	2×2	Leaky ReLU
8	Convolution	128@200×64	64@200×64	3×3	1×1	-	Leaky ReLU
9	Convolution	64@200×64	128@200×64	3×3	1×1	-	Leaky ReLU
10	Convolution	128@200×64	64@200×64	3×3	1×1	-	Leaky ReLU
11	Convolution	64@200×64	1@200×64	1×1	1×1	-	Sigmoid

Table 4. Network architecture of LargeNet. The outputs from layer 3 are concatenated channel wise to outputs of layer 12 to form the input to layer 13.

Layer #	Operation	Input Size	Output Size	Kernel Size	Pad Size	Pool Size	Activation
1	Convolution	1@1536×512	32@768×256	3×3	1×1	2×2	Leaky ReLU
2	Convolution	32@768×256	64@384×128	3×3	1×1	2×2	Leaky ReLU
3	Convolution	64@384×128	64@192×64	3×3	1×1	2×2	Leaky ReLU
4	Convolution	64@192×64	64@96×32	3×3	1×1	2×2	Leaky ReLU
5	Convolution	64@96×32	64@48×16	3×3	1×1	2×2	Leaky ReLU
6	Convolution	64@48×16	64@24×8	3×3	1×1	2×2	Leaky ReLU
7	Convolution	64@24×8	128@12×4	3×3	1×1	2×2	Leaky ReLU
8	Convolution	128@24×8	64@12×4	1×1	-	-	Leaky ReLU
9	Transposed convolution	64@12×4	64@24×8	4×4	1×1	-	Leaky ReLU
10	Transposed convolution	64@24×8	64@48×16	4×4	1×1	-	Leaky ReLU
11	Transposed convolution	64@48×16	64@92×32	4×4	1×1	-	Leaky ReLU
12	Transposed convolution	64@92×32	64@192×64	4×4	1×1	-	Leaky ReLU
13	Convolution	128@192×64	1@192×64	1×1	-	-	Sigmoid

contour finding on thresholded saliency maps. To the best of our knowledge, this approach is a first. This concept takes advantage of the unique nature of railway track images when compared to generic object detection challenges. FasteNet performs fastener detections on images of 1600×512 at a speed of 110 FPS on an Nvidia GTX 1080 with up to 99.1% precision using a very lightweight network.

As it stands, there are several limitations to FasteNet - susceptibility to perspective and size warping being a major one, limiting its use case to non-airborne mounted camera systems. FasteNet may also be overfitting to the dataset, given that the dataset is considered small, it may not fully capture the distribution of such an application in real life.

In the future, several things will be looked at. Firstly, there is hope to make the dataset public and contain more samples. Second, detecting fasteners alone is not particularly useful. FasteNet will aim to be applicable to fastener, sleeper, rail, and ballast defects by having one extra channel per class of detection. The goal is leverage the principles of FasteNet to develop similar models for railway inspection and/or road inspection. FasteNet can also be adapted to perform multitask classification as in the case of [5]. Additionally, an expansion of the dataset would allow the network to generalize better. Data augmentation methods such as a range of affine transforms, lux manipulation, self adversarial training or simply gathering more data can be used.

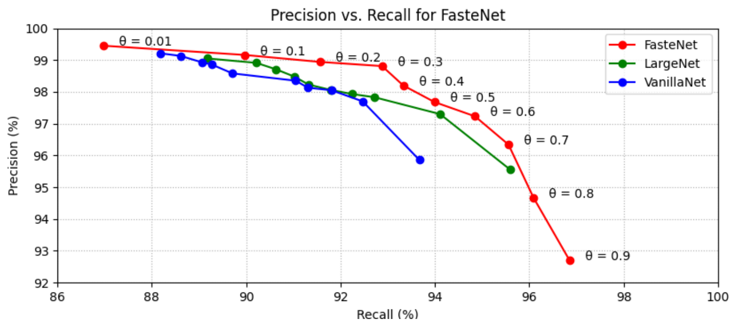


Fig. 3. Precision Recall Curve of Fastenet, VanillaNet, and LargeNet when performed on the unseen 297 images of the dataset with different threshold values.

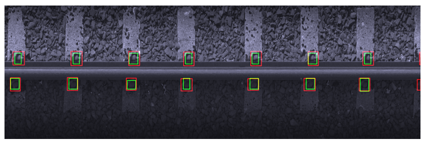
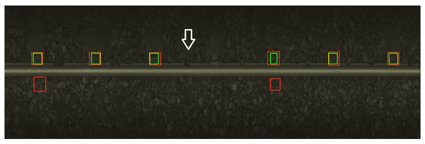
(a)  (b) 

Fig. 4. Green boxes are FasteNet predictions, red boxes are ground truth annotations. (a) 'False' Positives, the network predicts the presence of the fasteners even if small and not annotated in the ground truth. (b) False Negatives resulting from fastener occlusion and poor lighting conditions. Even so, the network hedges against detecting defect fasteners, identified with the white arrow here.

References

1. B. Marquis, J. LeBlanc, and A. Tajaddini, "Vehicle track interaction safety standards," in *ASME/IEEE Joint Rail Conference*, vol. 45356. American Society of Mechanical Engineers, 2014, p. V001T02A016.
2. Y. Xia, F. Xie, and Z. Jiang, "Broken railway fastener detection based on adaboost algorithm," in *2010 International Conference on Optoelectronics and Image Processing*, vol. 1. IEEE, 2010, pp. 313–316.
3. J. Liu, B. Li, Y. Xiong, B. He, and L. Li, "Integrating the symmetry image and improved sparse representation for railway fastener classification and defect recognition," *Mathematical Problems in Engineering*, vol. 2015, 2015.
4. X. Gibert, V. M. Patel, and R. Chellappa, "Robust fastener detection for autonomous visual railway track inspection," in *2015 IEEE Winter Conference on Applications of Computer Vision.* IEEE, 2015, pp. 694–701.
5. X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 1, pp. 153–164, 2016.
6. Q. Song, Y. Guo, L. Yang, J. Jiang, C. Liu, and M. Hu, "High-speed railway fastener detection and localization system," *arXiv preprint arXiv:1907.01141*, 2019.
7. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
8. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
9. T. Wang, F. Yang, and K.-L. Tsui, "Real-time detection of railway track component via one-stage deep learning networks," *Sensors*, vol. 20, no. 15, p. 4325, 2020.
10. C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.
11. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
12. A. Araujo, W. Norris, and J. Sim, "Computing receptive fields of convolutional neural networks," *Distill*, vol. 4, no. 11, p. e21, 2019.
13. J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
14. S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.
15. S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.
16. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
17. J. Liu, Y. Huang, Q. Zou, M. Tian, S. Wang, X. Zhao, P. Dai, and S. Ren, "Learning visual similarity for inspecting defective railway fasteners," *IEEE Sensors Journal*, vol. 19, no. 16, pp. 6844–6857, 2019.
18. X. Wei, Z. Yang, Y. Liu, D. Wei, L. Jia, and Y. Li, "Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 66–81, 2019.