Optimizing Neural Networks using Transfer Learning

Rehan Hafeez

The University of British Columbia, NTCO, FLIR IIS

Introduction

FLIR's Firefly Deep Learning camera is a novel device that can be preloaded with neural networks to independently make complex decisions and automate tasks such as quality control.

Image 1: FLIR's Firely DL Camera

The goal of this project was to determine if the preloaded networks could be optimized for more specialized tasks to reflect a user-case scenario with access to limited data. The problem was framed within the context of detecting defects in Printed Circuit Boards (PCBs), using FLIR's in house production line as our reference data source. Using a combination of transfer learning and data preprocessing we were able to show an increase in performance compared to the available generic models, with promising potential for further improvement.

Transfer Learning and Data Preparation

Transfer learning describes the process of taking a pre-trained network (such as Mobilenet_V1) and repurposing it for a more specialized task. The
rationale is that a large pre-trained network will act as a generic model of the visual world, and the weights in such a model will be optimized for general feature extraction. Transfer learning preserves these optimized weights, and therefore the ability to extract general features, while adapting the final classification layer to a more specific task which in our case is PCB defect detection. In order to do this, the classification layer must be retrained on data from the target domain.

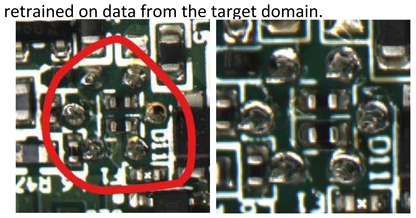


Image 2: Sample of "bad" PCB image (left) and "good" (right)

Fine-tuning is a subset of transfer learning, and involves "opening up" more than just the final classification layer to be retrained on data from the new target domain. In this process, minor adjustments to the weights in these layers can yield significant improvements in performance.

Using FLIR's in house PCB images, there are three possible regions of interest, with an example of one shown above. The first phase of the problem was allocated to manually sorting through these images, and assigning them to a "good" or "bad" class depending on the quality of the soldered joints. As this was project was a proof of concept test for optimization, the problem was simplified to a two-class dataset, which incorporated samples from two of the three regions of interest. The in-house trials conducted in the next section use this data to perform transfer learning on the Mobilenet_V1 model with the goal of classifying PCB images as "good" or "bad".

Neural Network Architecture

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Initial Trials using FLIR PCB Dataset

Trial	Loss	Accuracy (%)	Precision_0 (%)	Precision_1 (%)	Recall_0 (%)	Precision_1 (%)
Initial	2.64E-4	99.5	99.5	x	1	x
Forced Balance	0.25	95.0	91.2	x	1	x
Initial Class Weighted	0.01	99.74	91	99.8	78.1	99.9

Table 2: Initial Results on FLIR PCB data

The final dataset that the model was trained on contained a total of 9870 images split into two classes. This data was heavily imbalanced however, with only 120 "bad" images. This produced deceptively accurate initial results, which can be seen in the tabulated values below. With imbalanced data, a deep neural network will start to learn to expect mostly "good" values because there are disproportionately more positive samples, and during training this is reported as high accuracy. These models fail to generalize well when tested against new data. The first solution to this problem was to attempt to forcibly balance the data by reducing the number of positive samples to 600. This however worsened the problem with having limited data, and was not further pursued.

The final solution implemented class weights into the loss function, effectively penalizing the model far more severely for errors related to the under represented class. In order to better understand and track the behaviour of the model, class specific metrics were introduced for Recall and Precision. Precision describes the proportion of positive identifications that were actually correct, while Recall describes the proportion of positive samples that were correctly identified.

Optimization Trials using Missing Hole Dataset

In a user case scenario, a consumer would take the generic Mobilenet_V1 model, and use that network as the basis for transfer learning for their target domain. To simulate such a scenario, we used a subset of a publicly available PCB dataset (n/2=495) as our target domain for the optimization trials. This dataset shall be referred to as the Missing Hole dataset in the following discussion. Defects in these samples (not shown) consisted of missing soldered holes, and also formed a two-class classification problem.

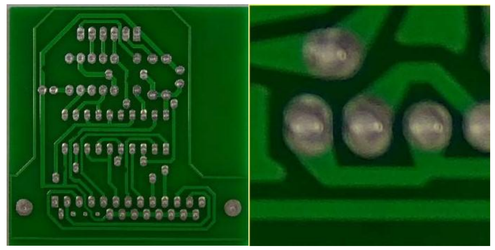


Image 3: Missing hole dataset sample (left) and cropped region of interest (right)

The final testing approach consisted of performing transfer learning on the Missing Hole dataset, using the Mobilenet_V1 network as the base model.
This trial was then used as a benchmark to test the performance of models trained on the missing hole dataset, but based on FLIR's PCB models and not Mobilenet.

The in-house trials conducted using FLIR's PCB data are intended to replace Mobilenet_V1 as the base model for transfer learning to the Missing Hole target domain. To that end, two main configurations were tested that differed on how many layers were fine tuned with FLIR's PCB data. The first configuration (t1) retrained the classification layer and the 13th layer of the model, while the second configuration (t2) opened up to the 11th layer of the model.

Results

The focus of the Optimization Trials shifted to the final performance of the models in question, and thus the results are framed in terms of the Accuracy of the Training and Validation split on the missing hole dataset. The total number of samples used was 990 with an equal number of "good" and "bad" samples. There was an 80% split to Training, and 10% to Validation. Note that the remaining 10% was allotted to a testing split that remained unused.

Trial	Training Accuracy (%)	Validation Accuracy (%)
Mobilenet_V1 Base	91.58	84.38
T1	92.53	84.54
T2	92.35	84.38

Table 3: Final results of Optimization Trials on missing hole dataset

Conclusion and Future Scope

Our findings show that a specialized network can outperform a generic model when adapted to a specific target domain. As a proof of concept these results are encouraging, but it is unlikely that any of these models are robust enough to be used in a practical manufacturing or quality control setting. Much higher accuracies would be needed to automate such tasks, and there remains potential for additional hyper-parameters in these models to be optimized.

Though not discussed in this report, there were extensive preprocessing functions applied to the data before training. Previous work has shown that variation in preprocessing functions can have a significant impact on the final model's ability to generalize to new or unseen data. Moreover, previous studies at FLIR have suggested that cropping the Mobilenet_V1 architecture can reduce inference time and maintain high levels of accuracy. The combination of cropping and transfer learning could be another potential source of improvement for these models. Finally, as with any machine learning process, training the specialized network on more abundant and varied data would likely increase the performance of these models drastically.

Acknowledgements

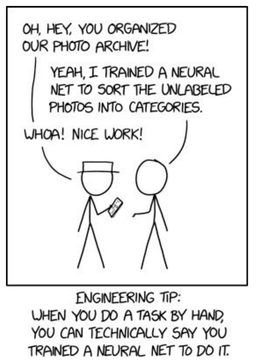
I would like to acknowledge and thank the Research Team at FLIR IIS, with special thanks to Di Xu, Stephen Se, and Ahmed Sigiuk for their guidance and contributions to this project. I would also to thank the NTCO program for providing the opportunity to be a part of this research.

References

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017, April 17). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

Huang, W., & Wei, P. (2019, January 24). A PCB Dataset for Defects Detection and Classification.

https://xkcd.com/2173/



NTCO

FLIR

THE UNIVERSITY OF BRITISH COLUMBIA