CS230

Detection and Classification of Defects in Parcel Packaging Using Deep Neural Networks

Hodges Haywood
SCPD
hhaywood@stanford.edu

Abstract

I investigate image classification on detecting and classifying structural damage to parcel packages that travel on an automated conveyor system. Currently, industrial sized automated parcel conveyance systems are manually monitored for the detection of damaged or defective packaging. Humans are not able to spot every damaged package on the many miles of conveyance belts. I investigate the challenge of using convolutional neural networks to classify the structural integrity of individual parcel packages on automated conveyance systems at various points during their journey. Convolutional neural networks have been successfully used in many practical areas, including defect and damage detection.

1 Introduction

Everyday, logistics and shipping companies lose millions of dollars because of damaged package claims from consumers. Statistics show that one out of every ten packages are lost or damaged[10]. This might not seem like much at first, but these companies ship millions of parcels per day. So, that amounts to hundreds of thousands damaged packages per day.
Often the contents of the parcels spill out onto the automated conveyor systems and become permanently damaged, or stuck, causing production flow stoppage. These packages might contain vital medicines, hazardous materials, hard metal objects, clothing, or even small weapons or ammunition that might become dislodged in the motor housings of the conveyor. Often times, the shipping labels are damaged along with the packaging and these packages need to be intercepted before they reach the label scanning cameras above the conveyors. Otherwise the system will not know where to divert packages with then become stuck in a continuous loop.
I investigate building a classification system that can detect whether a package is damaged or defective.

2 Related Work

Much work has been done in the areas of damage and defect detection and classification. Weimer, D., H.Thamer, B. Scholz et al. introduced a neural network which uses random generated image patches and statistical feature representations for defect detection on textured surfaces[1]. Essid Oumayma, Hamid laga, and Chafik Samir et al. have developed a machine vision framework for efficient detection and classification of manufacturing defects in metal boxes[7]. Their experiments on a database of real images have demonstrated that their work could out-perform the best models while remaining computationally competitive[7]. Wang, Teng, et al. used spatial distribution of potential pavement crack pixels for the detection of fissures in pavement whereby they could separate the cracks from pavement background. Their results showed that they could achieve higher detection as well as fewer false positives and false negatives compared to statistical learning based methods like Support Vector Machines(SVM)[10]. Fangzhou, et al. experimented with 3-dimensional deep neural networks to detect the presence of cancer nodules in lung tissue. Their results showed that using 3D neural networks effectively detected regions of malignant nodules[11]. Although this work is in the field of medicine, the application is the same; finding defects whereby the defects or damage are irregularly and randomly shaped sub-regions. My work differs from the previous approaches in that I did not use localization in the sub-regions of defect or damage. However this leaves much opportunity for improvements in future work where I experiment with different approaches to localizing defected and damaged regions.

3 Dataset

The dataset consists of 2000 training, 200 validation, and 200 test images. Half of the images are of boxes or packages with surface tears, holes of various size, or gashes. The other half of the images are of boxes or packages without defects. The data was collected from a Google search and from photographs taken at a logistics and shipping company.

3.1 Data Pre-processing

Pre-processing consisted of putting the data into three separate folders named training set, validation_set, and test_set. Each of these directories contain the two directories "damaged" and "un_damaged". I used the Keras **flow_images_from_directory** feature to turn these two directories into classes. The data are initially jpeg images. They are decoded into 3-channel(RGB) arrays of pixels and then converted into floating point tensors. The pixel values are then rescaled(normalized) by a factor of 1/255. The data is formatted into tensors prior to being fed into a neural network. Neural networks work better with pixel values that have been rescaled to the [0,1] interval[6].

4 Methods

All of the models used binary cross-entropy loss and sigmoid binary output. See equations (1) and (2).

4.1 Hardware and Software

The work was done using the keras framework with a Tensorflow backend. The models were trained on a desktop pc using a Nvidia Geforce 1080 8GB GPU.

4.2 Baseline Model

The baseline model consists of 5 convolution and maxpooling pairs, a flattened layer, a dense layer and a sigmoid output. This baseline was created in order to investigate how a simple model would perform.

4.3 Baseline Model with Augmentation and Dropout

I then added augmentation and dropout to the baseline model. Augmentation and dropout are used together to help reduce overfitting. I set dropout to be 0.5 and use width shift, height shift, rotation, and horizontal flip in the augmentation setting.

4.4 Pre-trained Models

The pre-trained models include VGG-16, ResNet-50, and Inception-v3. I removed the top fully connected layers from these models and froze the remaining convolutional layers. Freezing of the pre-trained model was done in order to preserve the weights. I then connected fully connected model with sigmoid outputs to the tops of the frozen layers.

4.4.1 VGG-16 Architecture

The VGG16 convnet was originally trained on 1.4 million labeled images and 1,000 different classes[6]. I felt this model was a good choice since it was trained on so many common images and that it might generalize well to the image data. VGG16 is also a good architecture from which to gain valuable intuition on how convolutional neural networks are built and how they work.

4.4.2 ResNet-50 Architecture

I chose ResNet-50 to investigate if its architecture can speed up training. ResNet-50 is a residual network that uses skip connections to add an early signal to the output of a layers much deeper into the network. This allows the model to make progress early on in the training.

4.4.3 Inception-v3 Architecture

The Inception-v3 architecture helps to reduce computational cost by introducing 1x1 convolutions to create bottlenecks in the network.

4.4 Relevant Equations

The binary cross-entropy loss function is expressed as:

$$J = -\sum_{i=1}^{N} y_i \log(h_\theta(x_i)) + (1 - y_i)\log(1 - h_\theta(x_i)) \quad (1)$$

The sigmoid output function is expressed as:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \quad (2)$$

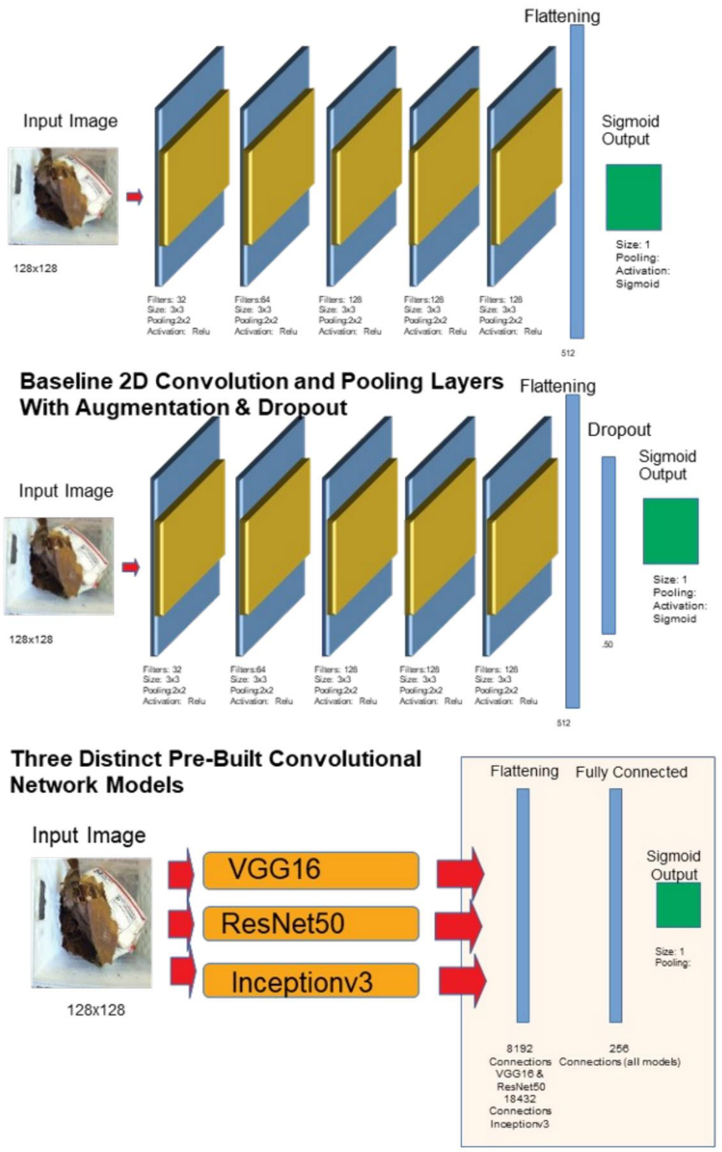

Figure 1. The baseline, baseline with augmentation and dropout, VGG-16, ResNet-50 and Inception-v3 models

5 Results and Discussion

Table 1 and **Figure 2** below both summarize the results and performance of the five models. The VGG-16 model had the best performance with 84% training accuracy and 88% validation accuracy. However validation accuracy should not be higher than training accuracy. This could be just a rare, random chance. The baseline model had the highest training accuracy of 99% with a validation accuracy of 80.5%. This indicates high variance and overfitting. The inception-v3 model had the lowest training performance of 77% with a validation accuracy of 67%. This would indicate high bias. With the exception of the ResNet-50 model, the validation accuracy of each model was not too far off from the training accuracy. The test accuracy each model was low compared to their validation training accuracy. The transfer learning models seemed to do better on test accuracy than the baseline models.

Model	Learn Rate	Epochs	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Test Accuracy	Transfer Learning
Baseline Model	0.0001	30	.015	1.02	99.5%	80.5%	25%	No
Baseline plus Augmentation and Dropout	0.0001	30	0.28	0.42	87%	83%	35%	No
VGG16	0.0001	30	0.36	0.37	84%	88%	48%	Yes
ResNet50	0.0001	60	0.23	2.17	89%	50%	50%	Yes
Inceptionv3	0.0001	30	0.50	2.98	77%	67%	50%	Yes

Table 1. Results from training and testing

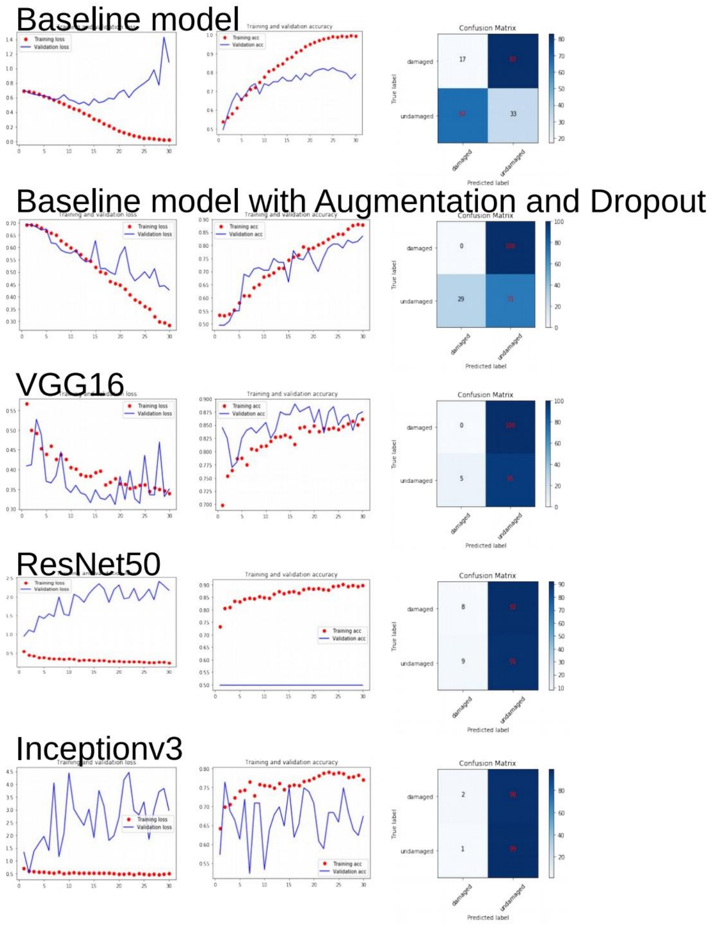

Figure 2. Graphical view of results along with confusion matrices

5.1 Visual Analysis

Visual analysis of both baseline models showed no specific activations in any of the layers although the appearance of the channels in each layer were different between the two models. As expected the regions of damage in the regions of damage showed very little activation. Figure 3 illustrates the visualizations of the layers of both models. With further work, the regions of defect or damage could be activated if the models were to be trained on appropriately labeled images, where the regions of damage are manually localized, bounded, and annotated[1][10][11]. See **Figure 3**

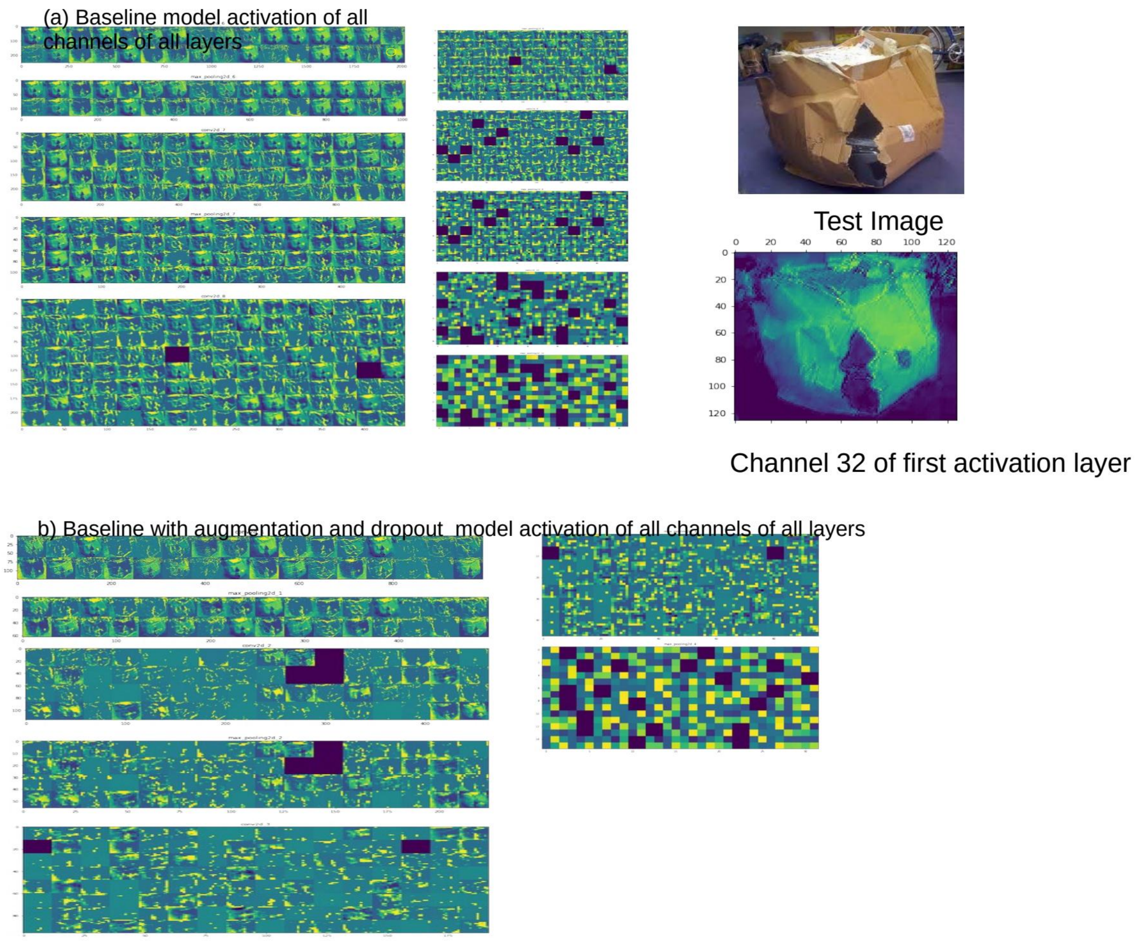

Figure 3: Visualization of intermediate layers for baseline model and baseline model with augmentation and dropout

6 Conclusion and Future Work

Despite having used five different models including three pre-trained architectures, all of the models performed poorly on the test data. Visualization of intermediate layers show that there are no specific activations, particularly in the zones of damage or defect. This is most likely due to the models not being able to separate background noise in the image, the background being the areas outside the regions of damage or defect. If the data had been trained on images where the regions of damage or defects were manually localized and surround with bounding boxes or circles and then annotated, much better performance results would have be attained. This leaves much room for improvement in future work where I experiment with different approaches to localizing defect and damage regions such as the method proposed by[7]. One such approach is to pre-process the training data by manually localizing the regions of defect. This should lead to better performance. Subsequent visualiztion of the convolutional layers should show activations in these localized regions.

Link to code can be found at this github link: https://github.com/usojourn/StanfordCS230

References

[1] Weimer, D., H. Thamer, and B. Scholz-Reiter. "Learning defect classifiers for textured surfaces using neural networks and statistical feature representations." *Procedia CIRP* 7 (2013): 347-352.

[2] Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

[3] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[4] Bakhary, Norhisham, Hong Hao, and Andrew J. Deeks. "Structure damage detection using neural network with multi-stage substructuring." *Advances in Structural Engineering*13.1 (2010): 95-110.

[5] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

[6] Chollet, Francois. *Deep learning with python*. Manning Publications Co., 2017.

[7] Essid, Oumayma, Hamid Laga, and Chafik Samir. "Automatic detection and classification of manufacturing defects in metal boxes using deep neural networks." *PloS one* 13.11 (2018): e0203192.

[8] Gopalakrishnan, Kasthurirangan, et al. "Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection." *Construction and Building Materials* 157 (2017): 322-330.

[9] Géron, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

[10] Wang, Teng, et al. "Automated shape-based pavement crack detection approach." *Transport* 33.3 (2018): 598-608.

[11] Liao, Fangzhou, et al. "Evaluate the Malignancy of Pulmonary Nodules Using the 3D Deep Leaky Noisy-or Network." *arXiv preprint arXiv:1711.08324* (2017).

[12] Sivaramakrishnan, Rajaraman, Sameer Antani, and Stefan Jaeger. "Visualizing deep learning activations for improved malaria cell classification." *Medical informatics and healthcare*. 2017.