AlphaNut: Nut/Screw Classifier via CNN
CS 230 Spring 2018

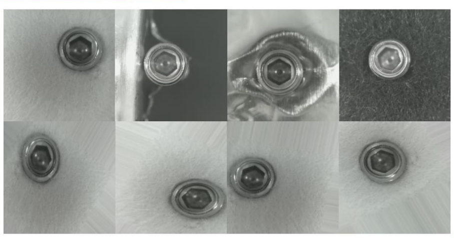
Sean Choi, Ryan Kwon
yo2seol@stanford.edu, ryankwon@stanford.edu

Introduction

- From new furniture, home appliances and more, the assembly and disassembly involves accurate **identification of nuts/screws**.
- Using convolutional neural network (CNN), a **precise and rapid identification of unknown screws** can be realized.
- To develop a platform to identify the screws, we **built our own dataset** by taking a fixed-distance images to contain the exact dimensional information.
- The output is a softmax prediction of the image to categorize the screw.

Data Set

- A **custom camera module** was built to measure the accurate size of the screws by **fixing the focal length** of every image.
- Screws can be found in diverse backgrounds / conditions and so to take that into consideration, we mimicked some possible situations as shown. Also for each image, we ran data **augmentation** (translation, rotation, zoom, flip, and shear) .
- Each screws were purchased with size specifications to ensure the ground truth.
- Our final dataset includes 491 taken photos, augmented and divided into **12036 training and 1263 test sets**.



Features & Model

	VGG16 [8]	VGG19 [8]	Resnet	InceptionV3	Inception-Resnet-V2	Mobilenet [6]	Xception[2]
Output Size	4096	4096	2048	51200	38400	50176	2045

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

- Features of our models consist of the weights obtained from passing our training dataset into the pre-trained models that are built from ImagenNet database. The feature sizes vary between which pre-trained models we used. For example, VGG16 and VGG19 outputs a weight vector of size 4096, which then becomes the feature of our final layer of the model that we optimize for.

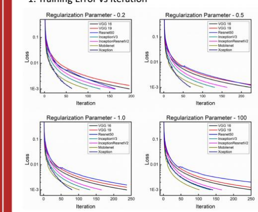
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

- We tested multiple models listed and compared the results for our application. Models were employed via **transfer learning**, built upon the weights obtained from various well-known CNN models and pretrained using ImageNet. Using transfer learning allowed us to make accurate predictions with limited dataset and avoid large computational power / lengthy training time.
- Given the pre-trained weights, we used **softmax regression** with various hyper parameters such as types of regularization, regularization parameter, and threshold criteria. The loss function is as follows

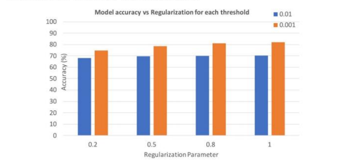
$$||\theta||_2^2 + C\sum_{i=1}^{n}\log\prod_{l=1}^{k}\left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}\right)^{1\{y^{(i)}=l\}}$$

Results & Error Analysis

1. Training Error vs Iteration



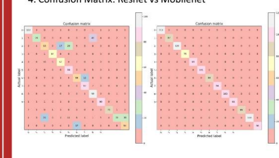
2. Resnet Model Accuracy per Regularization



3. Build Time

Regularization Type	VGG16	VGG19	Resnet	Inception V3	Inception-Resnet-V2	Mobile net	Xception
L2	59s	73s	33s	1459s	814s	842s	17s
L1	173s	209s	83s	1691s	1569s	1213s	79s

4. Confusion Matrix: Resnet vs Mobilenet



Discussion: From our training sets, the pre-trained weights and softmax regression loss function, we first looked at the training error vs iteration for each model. We fixed the learning rate α but looked through different regularization parameter. Each model, as expected, also exhibited a significant difference in the build time using L1 and L2 regularization. Overall, considering the build time and accuracy, we think **xception** provides an excellent choice for our application. Looking at the confusion matrix, we compared the worst to the best accuracy results. We hypothesize the high accuracy derives from our lack of data set and is overfitting the data. To compensate for this, we need more datasets and diverse types of test images.

5. Overall Model Accuracies

Threshold, Regularization	vgg16	vgg19	resnet50	inception-v3	inception-resnet-v2	xception	mobilenet
0.01, L1, C=0.2	98.97, 0.99	98.42, 0.98	69.60. 0.68	99.37, 0.99	99.52, 1.0	99.60, 1.0	99.92, 1.0
0.01, L1, C=0.5	99.52, 1.0	98.81, 0.99	73.24, 0.72	99.37, 0.99	99.52, 1.0	99.60, 1.0	99.92, 1.0
0.01, L1, C=0.8	99.52. 1.0	98.89, 0.99	74.35, 0.73	99.37, 0.99	99.52, 1.0	99.60, 1.0	99.92, 1.0
0.01, L1, C=1	99.52, 1.0	98.89, 0.99	74.58, 0.73	99.37, 0.99	99.52, 1.0	99.60, 1.0	99.92, 1.0
0.01, L2, C=0.2	99.21, 0.99	98.57, 0.98	62.55, 0.66	99.45, 0.99	99.52, 0.99	99.60, 1.0	99.92, 1.0
0.01, L2, C=0.5	99.21, 0.99	98.57, 0.98	68.17, 0.68	99.45, 0.99	99.52, 0.99	99.60, 1.0	99.92, 1.0
0.01, L2, C=0.8	99.21, 0.99	98.57, 0.98	69.99, 0.68	99.45, 0.99	99.52, 0.99	99.60, 1.0	99.92, 1.0
0.01, L2, C=1	99.21, 0.99	98.57, 0.98	70.23, 0.68	99.45, 0.99	99.52, 0.99	99.60, 1.0	99.92, 1.0
0.001, L2, C=0.2	99.6, 1.0	99.29, 0.99	74.66, 0.73	99.6, 1.0	99.6, 1.0	99.76, 1.0	99.92, 1.0
0.001, L2, C=0.5	99.6, 1.0	99.29, 0.99	78.7, 0.78	99.6, 1.0	99.6, 1.0	99.76, 1.0	99.92, 1.0
0.001, L2, C=0.8	99.6, 1.0	99.29, 0.99	81.08, 0.80	99.6, 1.0	99.6, 1.0	99.76, 1.0	99.92, 1.0
0.001, L2, C=1	99.6, 1.0	99.29, 0.99	82.03, 0.81	99.6, 1.0	99.6, 1.0	99.76, 1.0	99.92, 1.0

Future Works

- Expand both the test and training data by acquiring more broad dataset of diverse nut/screw types.
- Expand AlphaNUT to real-time/simple application on mobile platforms.
- More fine-tuning of our models to achieve lower error on new test data set.
- Look into augmenting test data set and using both obtained and augmented data for classification.

References

[1] A. Alemi. Improving inception and imaging classification in tensorflow, 08 2016.
[2] F. Chollet, Xception: Deep learning with depthwise separable convolutuions. *CoRR*, abs/1610.02357, 2016.
[3] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646-1654, 2014.
[4] Google. Google colab. URL https://colab.research.google.com/
[5] Y. Guo, Y. Liu, A. Oerlemans, S.-Y. Lao, S. Wu, and M. S. Lew, Deep learning for visual understanding: A review. 187, 11 2015.
[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andereetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.
[7] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature,* 521:436 EP-, 05 2015
[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.