2022 26th International Conference on Pattern Recognition (ICPR)
August 21-25, 2022, Montréal, Québec, Canada

Towards Understanding the Behaviors of Pretrained Compressed Convolutional Models

Timothy Zee
Department of Computer Science
Rochester Institute of Technology
Rochester, New York
Email: tsz2759@rit.edu

Manohar Lakshmana
Department of Computer Science
Rochester Institute of Technology
Rochester, New York
Email: mxv5801@rit.edu

Ifeoma Nwogu
Department of Computer Science
University at Buffalo, SUNY
Buffalo, New York
Email: inwogu@buffalo.edu

2022 26th International Conference on Pattern Recognition (ICPR) | 978-1-6654-9062-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICPR56361.2022.9956037

***Abstract*—We investigate the behaviors that compressed convolutional models exhibit for two key areas within AI trust: (i) the ability for a model to be explained and (ii) its ability to be robust to adversarial attacks. While compression is known to shrink model size and decrease inference time, other properties of compression are not as well studied. We employ several compression methods on benchmark datasets, including ImageNet, to study how compression affects the convolutional aspects of an image model. We investigate explainability by studying how well compressed convolutional models can extract visual features with t-SNE, as well as visualizing localization ability of our models with class activation maps. We show that even with significantly compressed models, vital explainability is preserved and even enhanced. We find with applying the Carlini & Wagner attack algorithm on our compressed models, robustness is maintained and some forms of compression make attack more difficult or time-consuming.**

I. Introduction

Small technological devices are becoming more capable of creating high resolution images and videos, which are routinely uploaded and downloaded to and from the cloud. For example, the resolution of a single image taken by a standard household security camera such as Ring®is 1920-by-1080 full high definition (fHD).

But to use such images/videos in practical applications, such as in an algorithm to detect a specific moving object, an efficient solution would be to process the data locally on a household device such as a phone or small computer, using a compact machine learning architecture. Similar examples can be made for creating intelligent applications for handling personal images and videos on smart phones and similar devices. Hence, there has been an urgent need for more compact, compressed architectures with small enough footprint to be run on smaller devices.

As neural networks continue to enter new domains, their model structures and hardware requirements continue to increase. However, advancing neural network architectures results in expanding in model capacities, thus making them intractable for many small devices and mobile applications. Techniques to compress neural networks to run on embedded hardware is an active area of research. However, the properties provided by compression techniques to neural models are not well understood. Particularly, key properties such as model robustness and explainability are not yet well understood.

In this work, therefore, we evaluate multiple compression methods to better understand their behaviors when applied to pretrained convolutional models. We accomplish this by (i) observing changes in model performance after compression in applied; (ii) investigating the resistance of the various compressed models to adversarial attacks; and (iii) lastly, discerning their explainability properties after the convolutional models are compressed.

II. Related Work

Understanding how compression affects a neural network model has been studied by many researchers in the machine learning field and most of the techniques fall in to one or more of the following categories: (i) pruning; (ii) quantization; (iii) sparsity and low-rank approximation (iv) distillation and (v) automated architecture engineering. The first two techniques do not require the original baseline network to be retrained, whereas the last three require some form of retraining.

One of the earlier works in weight pruning by Han et al. [10] successfully reduced the network weight parameter by almost a factor of 10 and with negligible loss in accuracy using heuristics and iterative weight pruning. Other pruning methods [5], [26], [6] were determined to be more selective in their pruning techniques. Because weights are pruned arbitrarily, these non-structured pruning methods result in irregular, sparse matrices making it hard to parallelize the network implementation. More structured pruning techniques involve filter pruning, channel pruning, and filter shape pruning [24], [13]. The resulting structured matrices were now full and regular, overcoming the implementation limitations of the non-structured methods. In a recent paper, Ma et al. [14] demonstrated that structured pruning methods are more preferable than non-structured ones in terms of storage overhead and computational efficiency.

While pruning involves reducing the number of weights in a network, weight quantization involves reducing the size of those weights. This can be accomplished by reducing the range of the weight values from a larger to a smaller set. Neural networks are by nature superfluous and over-parameterized, hence there is significant redundancies in the models. One of the earlier works in quantized neural networks by Vanhoucke et al. [21] involved a fixed-point implementation

using 8-bit integer activations. Anwer et al. [2] quantized the parameters of a pretrained network, layer by layer, using L_2 error minimization, and retrained the entire network with the quantized weights. They tested on MNIST and CIFAR-10, and showed that quantization improved generalization, giving higher classification results than the high precision networks, while also reducing the memory storage by about one-tenth. Other approaches explored different forms of quantization including Huffman codes, binning parameters into buckets, grouping connection weights with a hash function and vector quantization of the fully connected layers [11], [4], [7]. Other works in quantization include [25], [23].

Other forms of compression involve low-rank factorization of convolutional filters where full-rank feature matrices are factorized into single vectors (to be linearly combined), due to the separability property of convolution [15]. Similarly, *distillation* is a process where "knowledge" is transferred from a large trained model to a smaller one by training it to mimic the outputs of the larger model [14], [22].

Although we have discuss different compression paradigms in this section, for the rest of the paper, we take a deep dive into how standard convolution models behave when compressed with pruning and quantization methods specifically.

III. Compression Methods of Interest

To analyze how compression affects convolutional models, we employ pruning and quantization, two of the most popular methods for model compression. In addition, to see how well the observed behavior generalizes, we create a custom layer-ablated model to simulate the effects of extreme compression on a convolutional model. We also combine compression methods for analysis with a pruning+quantization model. Each method is discussed in further detail in the following sections.

A. Pruning

Pruning allows for ranking feature importance and cutting synapses, nodes, or even whole filters from the layers of a model. Li et al. [16] provide an implementation where the Hessian of the parameters are taken along with the loss function to find parameters that can be removed with little to no degradation in model performance. Pruning can be completed by ranking filters, $\hat{\mathbf{F}}$, and removing filters $\hat{F}_i$: $\hat{F_{i+n}}$ until a threshold is reached:

$$\hat{\mathbf{F}^l} = \{\hat{F}_i, \hat{F_{i+1}}, ..., \hat{F}_n\} = sort\big(\sum_{i=0}^{f_{in}} \sum_{j=0}^{f_{out}} |F_{ij}^l|_{L_1}\big) \quad (1)$$

$$\sum_{acc \geq t} \sum_{i=0}^{N} \big(\hat{F}_i^l \otimes 0\big) \quad (2)$$

Filters that have a low L1 rank are assumed to not provide useful feature extraction in a model, and removal of these filters does not significantly hurt performance. The same is true from cutting synapses or nodes which have low ranks as they are not helping shape predictions compared to high ranking filters or nodes. Pruning filters can be done in multiple ways such as removing entire filter blocks, removing entire channels of filters, or only pruning element-wise in individual feature maps. Pruning also varies with pruning for fully-connected layers, which commonly is done by either removing entire nodes, or cutting individual synapses between nodes. In this work, we prune per-channel and per-synapse, allowing for more flexibility over dropping entire filters or nodes.

B. Quantization

Quantization aims to shrink the size of the learnable parameters themselves, rather than cutting specific features of a model. Typically, libraries like Tensorflow [1] and PyTorch [18] use 32 bits of precision to represent weights in a model. Quantization aims to reduce this representation significantly to a smaller size such as 8 bits. To reduce model size without degrading performance, quantization re-normalizes all weights to the extreme bounds of the new representation size (e.g. 8 bits of precision re-normalizes to [0, 255]), allowing to maximize the fidelity the new parameters can represent. Quantization in 8 bits is shown in Algorithm 1 where the model weight's minimum and maximum, wMin and wMax, are used in conjunction with the re-normalization extremes, qMin and qMax, to quantize and dequantize model weights with the scale factor, σ, and zero point μ.

Algorithm 1 Methodology to quantize and dequantize a model's layer weights to 8-bits.

```
#Quantization
for l in model.layers do
    wMin = min(l.weights); wMax = max(l.weights)
    qMin = 0; qMax = 255
    σ[l] = (wMax - wMin) / (qMax - qMin)
    μ[l] = (qMin - wMin) / σ[l]
    qWeights[l] = (data + μ[l]) / σ[l]
end for
#Dequantization
for l in model.layers do
    deqWeights[l] = (qWeights[l] - μ[l])
end for
```

C. Ablation

To further explore the behaviors of compression on convolutional models, we create a layer-ablated model that discards most of the class-specific and least generalized model features. While pruning is focused on cutting out model parameters, it does so only second to preserving model accuracy. Pruning is effective for removing unnecessary, over-saturated, or even duplicated features while doing so with preserving high model performance. With the ablation model, the goal is instead to specifically target the most class-specific features and remove them, forcing the model to use the most generalized convolutional features during inference. This is completed by cutting entire convolutional layers, rather than cutting select filters like in pruning. We create the layer-ablated model to simulate over

compressing convolutional layers to help better understand compression's behavior on CNNs.

To create the ablated model, we follow the steps in Algorithm 2, where θ and $\hat{\theta}$ represent the parameters and best found parameters for the network, Net(X, θ), optimized by minimizing loss $\mathcal{L}$ with labels Y. Ablating M layers removes the last $N - M$ convolutional layers from the model and once all other layers are frozen, recovery training only helps the fully-connected layers to learn to make the best use of the more generic extracted filters.

Algorithm 2 Methodology to ablate a CNN's N-M convolutional layers.

Returns CNN that uses the $1 : (N-M)$ convolutional layers during inference.
for e in EPOCHS **do**
  $\hat{\theta}$ = min($\mathcal{L}$) s.t. (Net(X, θ), Y)
end for
for l in Net($\hat{\theta}$).layers **do**
  if l.dims == 4 **and** l $\geq$ (N-M) **then**
    l.ablate()
  else
    l.freeze()
  end if
end for
if l.dims == 2 **and** l.FIRST_FC **then**
  l.reshape(flatten((l-1).outputs)
end if
for r in RECOVERY_EPOCHS **do**
  $\hat{\theta}$ = min($\mathcal{L}$) s.t. (Net(X, θ), Y)
end for

D. Pruning+Quantization

In our exploration of compressed convolutional behavior, we also consider the case of multiple methods of compression applied to the same model by creating a pruned+quantized model. We first apply the previously mentioned pruning approach until a decreased performance threshold is reached, and then applying the resulting network to 8-bit quantization.

IV. Experiments, Results and Discussions

A. Model Architecture and Data

We build a VGG-style convolutional model for our experiments. We use six convolutional layers to expand channel-wise and therefore refer to this model as "VGG-6". We convolve with small filters using $[3 \times 3]$ filters for each layer of convolution. Max pooling is used in every other layer starting in the second layer to reduce dimensionality by cutting width and height by half. Batch normalization is applied after the convolution of every layer. After propagating through the six convolutional layers, the model flattens data and passes it through three fully-connected layers, flattening down to 2048 by 1024, 1024 by 512, and 512 to 10 with a SoftMax applied to give normalized predictions at the output of the model.

We train these models on three standard benchmark datasets, SVHN, CIFAR-10, and ILSVRC2012 (ImageNet). SVHN and CIFAR-10 are standard RGB color datasets we use in evaluating our VGG-6 model, while ImageNet allows us to study the effects of compression as we scale up model and data. To evaluate on ImageNet we apply the standard VGG-16 model with batch normalization completed between each layer.

B. Performance Metrics

Table II shows the performance results obtained from applying the different compression techniques on our three evaluation datasets. These results are discussed in detail in Section IV-B3.

1) Our VGG-6 pruning and quantization: We apply both pruning and quantization for our tests as they are the most widely used compression models. Pruning cuts out unnecessary weights and forces the model to re-use weights and make the most use out of them. We prune varying percentages of weights to retain a model performance accuracy to about a 5% performance drop. We pruned all layers except the first convolutional layers as they have been shown to be the most vital to model performance [9]. This allowed us to shrink the overall model (with the exception of the first convolutional layers) by the following: 40% for SVHN, 20% for CIFAR-10, and 20% for ImageNet. We allow 10 epochs, 20 epochs, and 9 epochs of recovery training after pruning respectively.

We also apply quantization, a method that shrinks the size of weights and biases significantly. Our quantization implementation takes weights and biases, represented as datatype Float32 to 8 bits. In PyTorch, Float32 variables can represent the range of $[-2^{31}, 2^{31}]$, whereas by compressing with quantization to 8 unsigned bits, the new range becomes [0, 255]. Re-normalization is then computed to fit to the new scale. At inference, the range, i.e, (min, max) of all floating-point tensors in the model are rescaled to improve latency.

2) Our VGG-6 Ablation: We create the layer-ablated model, shown in Figure 1 by first training our VGG-6 network to convergence. We then cut the last convolutional layer and reshape first fully-connected layer to fit the correct flattened dimensionality of the output of the new last convolution layer. Retraining is now done only on the fully-connected layers as they have been interrupted with this dimensionality change, however all convolutional layers remain frozen. Retraining only the fully-connected layer allows it to best use these more generic filters to provide optimal performance. The new inference pass of this model now uses the first $N - M$ layers of the model, forcing the model to use more generic features over the class-specific features learned in the ablated layer.

While we only ablate one layer of convolution, that ablated layer accounts for 16% of the total convolutional filters of the model, reducing the model significantly. While pruning intelligently removes unnecessary, over-saturated, or even duplicated features, we opt to blindly ablate the entirely layer of the most class-specific features to further study compression's effects for an over-dramatized model.

Data Set	Model	Accuracy	Model Size	Time
SVHN	Baseline	91.07%	9.87MB	5.566s
SVHN	Pruned	90.00%	9.87MB	5.180s
SVHN	Quantized	90.38%	2.79MB	5.298s
SVHN	Ablated	80.00%	8.11MB	9.740s
SVHN	Pruned+Quant8	90.20%	2.79MB	4.925s
CIFAR-10	Baseline	81.10%	9.87MB	4.823s
CIFAR-10	Pruned	74.80%	9.87MB	4.668s
CIFAR-10	Quantized	81.90%	2.79MB	4.638s
CIFAR-10	Ablated	75.00%	8.11MB	3.935s
CIFAR-10	Pruned+Quant8	75.70%	2.79MB	4.506s
ImageNet	Baseline	73.37%	540MB	2380.19s
ImageNet	Pruned	68.00%	540MB	2378.77s
ImageNet	Quantized	73.46%	178MB	2336.25s
ImageNet	Ablated	68.00%	531MB	2358.03s
ImageNet	Pruned+Quant8	68.40%	178MB	2307.82s

TABLE I: Compression evaluation of test accuracy, model size, and average inference time for SVHN, CIFAR-10, and ImageNet on the baseline, pruned, quantized, ablated, and pruned+quantized model where quantized is done at 8-bits and inference time is computed by taking the summation of inference time for 10,000 samples.

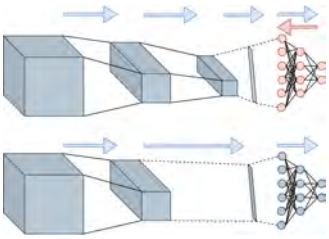

Fig. 1: Creating the layer-ablated model. The original model is trained end-to-end until convergence, the last convolutional layer is removed, and the first fully-connected layer's dimensionality is adjusted to fit the revised model. All the convolutional layers are then frozen, and the model is re-trained to convergence (top). Re-training the model now only affects the fully-connected section of the model. The inference pass now uses the frozen convolutional layers and the newly-trained fully-connected layers, making the model use the first $N - M$ convolutional layers of the baseline model (bottom).

3) Discussion on overall model performance: From the results presented in Table II, we observe that with implementing compression methods to retain within 5% of the baseline accuracy, the model size on the filesystem is reduced for all but the ablated compression model, and the quantized model has the most reduction. The size of the ablation variant increases only because of the structure of the VGG-6 model. When the top convolution layer is ablated, the next layer with its larger number of filters and feature maps is flattened, resulting in a significantly larger number of nodes for the new fully-connected layers. The deeper the original model, the less impact this type of ablation will have on the physical model size, as observed with VGG-16.

When the various models are run on different datasets, the inference times for 10,000 samples are reduced for all compression models, even when more than one compression technique is applied. In general, quantization improves performance and reduces overall model size on the filesystem. Similarly, Han et al. [10] showed that even when the sizes on file of the compressed and baseline models are similar, because of number of zero weights in the compressed model, the number of FLOPs is reduced significantly between the baseline models and their compressed variants. He et al. show that on embedded systems, there are even more significant latency gains that can be obtained with compressed models.

C. Explainability

To better understand the behavior that compression methods have when applied to convolutional models, we also study an increasingly important property of convolutional models, *model explainability*.

Explainability methods leverage convolution's ability to represent spatially important data in images. Because the convolution layers of the network preserve the spatial relationships among pixels, processes such as filter visualization, class activation maps (CAMs) [28], etc are effective methods for explainability. We review both t-SNE plots [20] and CAM results for two datasets in this section, and the ensuing results are discussed in detail in Section IV-C3.

1) T-distributed stochastic neighbor embedding (t-SNE) on ImageNet: Convolution layers allow the forward propagation of data to retain the pixel-level relationships with neighboring pixels. Once a model is flattened and passed to a fully-connected layer, valuable information is lost. Therefore, to understand how compression affect's convolutional models, we visualize t-SNE on the embeddings of the model up until the output of the first fully-connected layer of VGG-16. Accessing data at this level allows us access to embeddings generated predominantly by convolutional filters. We pick only ten classes from ImageNet to make the t-SNE visualization more human understandable. The selected class labels are: *trench, great white shark, bald eagle, scorpion, aircraft carrier, ambulance, balloon, desktop computer, gondola, table lamp*. We apply the default t-SNE parameters to all compression models, without tuning for increased performance.

The 2-dimensional maps generated by t-SNE, visually provide some insight into the decision making processes of convolutional models as we examine the maps from the baseline and various compressed models. We are interested in observing the clustering tendencies from the t-SNE plots, to determine if convolution is still a strong feature extractor, even

(a) Baseline (b) Pruned (c) Quantized (d) Ablated (e) Pruned+Quantized

Fig. 2: t-SNE Visualizations of each compressed model from the VGG-16 model on ten classes from the ImageNet dataset.

after significant compression. If t-SNE shows strong clustering tendencies, this indicates that the convolutional aspects of the network still have strong discriminability, hence, the key requirements for explainability.

We generate t-SNE plots of the first fully-connected layer for the baseline, pruned, quantized, ablated, and pruned+quantized models and visualize them in Figure 2.

2) Class activation maps (CAMs): CAMs showcase the natural localization ability of convolutional models when trained for classification tasks. We not only test how various compression methods affect the feature extraction from convolutional layers (via t-SNE), but we also analyze how they affect the natural localization performance.

To accomplish this, we take the VGG-6 model, train it to fit the SVHN dataset, and then generate CAMs using guided back-propagation. CAMs provide a heat-map over the "focus" points of the network by traversing backwards through the network via guided back-propagation, to re-map the positive activations of the model back to the input space. From here, the strength of these activations can be visualized as a heat-map for easy understandability. We visualize the results from four models when applied on SVHN: for the baseline, pruned, quantized, ablated, and pruned+quantized models. The CAM results can be seen in Figure 5.

3) Discussion on t-SNE and CAM results:

a) T-SNE: From the t-SNE maps shown in Figure 2, applied on 10 classes fom ImageNet, we observe that no compression methods applied does worse than the baseline model. The quantized model shows the most distinct clusters compared to the other models and the pruned model shows more distinctive clusters than the baseline, probably due to the removal of redundant filters.

b) CAM: We observe CAM samples obtained for SVHN, the dataset selected for this test due to the distinct regions of interest containing digits in the image. Using CAMs, we expected the convolution model to focus mainly on the region containing the digits as well as any distinctive local regions distinguishing the particular digit being tested. As we observe and show in Figure 5, the baseline model does not put any strong emphasis on the region of the image containing the digit. Although it highlights some regions of interest, these are not intuitive for explainability. The quantized model consistently appears to perform as expected, focusing on the region of the digit and highlighting sub-regions in the image that can potentially distinguish one digit from another. For example, on the second row (containing 2), the region of the digit has strong heat-map along the areas of curvature (distinguishing regions). From an explainability perspective, this result is closer to what we expected. The pruned and ablated models also behave as expected although not as definitive as the quantized model.

D. Robustness

In addition to exploring how explainability is affected with compression, we elect to study how robustness, another key area in modern networks fairs with compression.

To analyze how compression methods affect a model's sensitivity to adversarial attacks, we apply the Carnili and Wagner (C&W) L2 adversary attack [3]. C&W is considered the standard benchmark for fooling samples in neural networks. It works by not only minimizing the original class accuracy, but aims to increase accuracy in all other classes, creating a significantly stronger attack than the traditional fooling method like in [8]. The traditional method only attempts to maximize a target class so that the neural network will not predict the original class, whereas C&W works on all other classes.

We train VGG-6 model with C&W L2 objective function and then apply each compression technique; we then analyze the baseline, pruned, quantized and pruned+quantized variations. To evaluate robustness, we randomly select 1000 samples from CIFAR-10 test dataset and create adversarial samples by running the C&W L2 attack for 40 iterations. The mathematical representation for C&W is shown in Equations 3 and 4.

$$\begin{aligned} &while\ (\sigma(X' < \tau)): && \mid \lambda + (1 - X) \mid_{L2} + c(\lambda + 1) \\ &where: && \lambda = tanh(W) \in [-1, 1] \end{aligned} \tag{3}$$

$$f(X') = max(max(\sigma(X')_c, c \neq t) - \sigma(X')_c, -\tau) \tag{4}$$

The parameter τ is a confidence threshold, used as a termination condition, that analyzes output of the model on the adversarial data, $\sigma(X')$. w represents the change of variables for the equation $(X + \delta) = (1/2)tanh(W + 1)$ such that $tanh(.) \in [-1, 1]$. This change of variables is done because $-1 \leq (1/2)tanh(W + 1) \leq 1$ forcing $x + \delta$ to always be a valid solution with $0 \leq x + \delta \leq 1$. $f(x)$ is a chosen objective function which was empirically found that $f(X') = max(max(\sigma(X')_c, c \neq t) - \sigma(X')_c, -\tau)$ was found

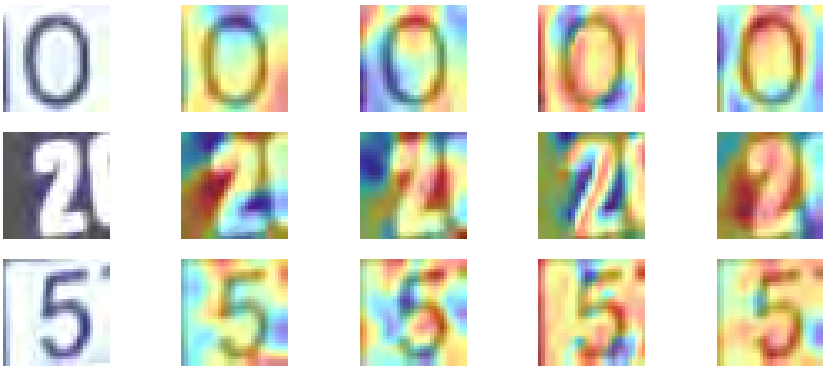

Fig. 5: SVHN class activation maps for selected samples on the original (far left) baseline (mid left), pruned (center), ablated (mid right), and quantized models (far right).

Model	Pre-fooled Acc	Post-fooled Acc	Pre-fooled conf	Post-fooled conf	Time (ms)
Baseline	81.10%	47.30%	98.16%	83.24%	8283ms
Pruned	74.80%	45.10%	95.30%	72.60%	5270ms
Quantized	81.90%	81.90%	98.04%	98.04%	1375ms
Pruned+Quantized	75.70%	75.70%	95.31%	95.31%	1222ms

TABLE II: Evaluation of foolability of the baseline, pruned, quantized, and pruned+quantized models on CIFAR10. Average pre- and post-fooled accuracy on the test set as well as confidence and average time to fool is recorded. Each test is run with 1000 generated samples.

best for the L2 attack. The constant c is set to the smallest value that performs best for $x*$ where $f(x*) \leq 0$ [3].

We first evaluate the average pre-fooled test accuracy and average pre-fooled confidence in predictions. After fooling via C&W, we re-evaluate the same 1000 samples to see how well the models were able to hold up to attack (i.e how strongly they naturally provide defensive capability against the attack). We also track the average time it takes to fool a sample from the compressed models. While C&W is known to generate adversarial examples extremely well, we are interested to see if compression can slow down its attack rate.

1) Discussion on C&W results: First, we find that none of the three compressed models have worse defense against C&W than the baseline model. Second, we find that when quantization is applied either by itself or after pruning, the C&W attack is not effective with the tested configuration. The baseline model average accuracy decreases by 33.8% with an average confidence drop of 14.92%. The pruned model handles the attack slightly better at an average decrease in performance of 29.7% and an average confidence drop of 22.7%. We find C&W runs about 1.57x quicker than the baseline, making a pruned model more susceptible to time sensitive attack.

Although the quantized model runs 40 iterations of C&W much quicker than the baseline, being 6.02x quicker, and 6.78x faster for pruned+quantized, it is irrelevant as the attack is ineffective in our tests. We use the same parameters for all tests, including an attack learning rate of 0.5e-4. To ensure quantization is not just sensitive to this learning rate, we re-test both the quantization and pruned+quantization model at attack learning rate in the range [1e-4, 0.5e-3] and find none of these rates have any effect (once 8-bit quantization is performed).

V. Conclusion

Compression provides AI access to new landscapes running on embedded or hardware constrained systems. While there has been research in methodologies, understanding how models behave under compression has not been as well studied.

In this work, we have studied behaviors of compressed convolutional models for two key areas, model explainability and robustness. We employ two popular compression techniques, pruning and quantization, along with a custom layer-ablated model and a combination pruning+quantization model to study these properties. We find via t-SNE maps that the essential component of explainability, convolutional feature distinguishability, is preserved among all compressed variants. Using CAMs, we find localization ability within convolutional models is preserved and even enhanced with compression.

To explore how compression affects robustness, we apply C&W and find that even with significant compression, robustness persists and in some cases can be used to help safeguard against such attacks. The empirical findings in this paper, especially relating to the success of quantization is consistent with findings in Han et al. [11].

Going forward, we are interested in analyzing convolutional model behavior on different types of hardware: CPU, GPU and mobile GPU. Because quantization is still hardware restricted, where gains can be tied to the hardware being used, it is important to benchmark the work against commonly used hardware and explore its limitations. Otherwise, given much of empirical evidence from this study, it is imperative to perform quantization (when possible) on large high-precision networks, even for basic tasks. We are interested in furthering the experiments conducted with model structures like those with residual [12] or inception [19] modules. In addition, we seek to expand the model defense study with compressed image models with algorithms such as those in [27] and [17].

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, and C. C. *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[2] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1131–1135.

[3] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[4] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International conference on machine learning*. PMLR, 2015, pp. 2285–2294.

[5] X. Dai, H. Yin, and N. K. Jha, "Nest: A neural network synthesis tool based on a grow-and-prune paradigm," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1487–1497, 2019.

[6] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Advances in Neural Information Processings and Systems (NeurIPS)*, 2017, pp. 4857–4867.

[7] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[9] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[10] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processings and Systems (NIPS)*, 2015, pp. 1135–1143.

[11] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016, pp. 1–14.

[12] 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.

[13] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[15] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[16] H. Li, J. Chen, H. Lu, and Z. Chi, "CNN for saliency detection with low-level feature integration," *Neurocomputing*, 2017.

[17] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *International Conference on Learning Representations*, 2019.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[20] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[21] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *=Deep Learning and Unsupervised Feature Learning NIPS Workshop, year=2011*.

[22] H. Wang, H. Zhao, X. Li, and X. Tan, "Progressive blockwise knowledge distillation for neural network acceleration." in *International Joint Conference on Artificial Intelligence IJCAI*, 2018, pp. 2769–2775.

[23] P. Wang, Q. Chen, X. He, and J. Cheng, "Towards accurate post-training network quantization via bit-split and stitching," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9847–9856.

[24] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," pp. 4857–4867, 2016.

[25] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4820–4828.

[26] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695.

[27] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[28] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.