Demonstration of Krypton: Optimized CNN Inference for Occlusion-based Deep CNN Explanations

Allen Ordookhanians
University of California
San Diego
aordookh@eng.ucsd.edu

Xin Li
University of California
San Diego
xli222@eng.ucsd.edu

Supun Nakandala
University of California
San Diego
snakanda@eng.ucsd.edu

Arun Kumar
University of California
San Diego
arunkk@eng.ucsd.edu

ABSTRACT

In this demonstration, we present KRYPTON, a system for accelerating occlusion-based deep convolution neural network (CNN) explanation workloads. Driven by the success of CNNs in image understanding tasks, there is growing adoption of CNNs in various domains, including high stakes applications such as radiology. However, users of such applications often seek an "explanation" for why a CNN predicted a certain label. One of the most widely used approaches for explaining CNN predictions is the occlusion-based explanation (OBE) method. This approach is computationally expensive due to the large number of re-inference requests produced. KRYPTON reduces the runtime of OBE by up to 35x by enabling incremental and approximate inference optimizations that are inspired by classical database query optimization techniques. We allow the audience to interactively diagnose CNN predictions from several use cases, including radiology and natural images. A short video of our demonstration can be found here: `https://youtu.be/1OWddbd4n6Y`

PVLDB Reference Format:
Allen Ordookhanians, Xin Li, Supun Nakandala, Arun Kumar. Demonstration of Krypton: Optimized CNN Inference for Occlusion-based Deep CNN Explanations. *PVLDB*, 12(12): 1894-1897, 2019.
DOI: https://doi.org/10.14778/3352063.3352093

1. INTRODUCTION

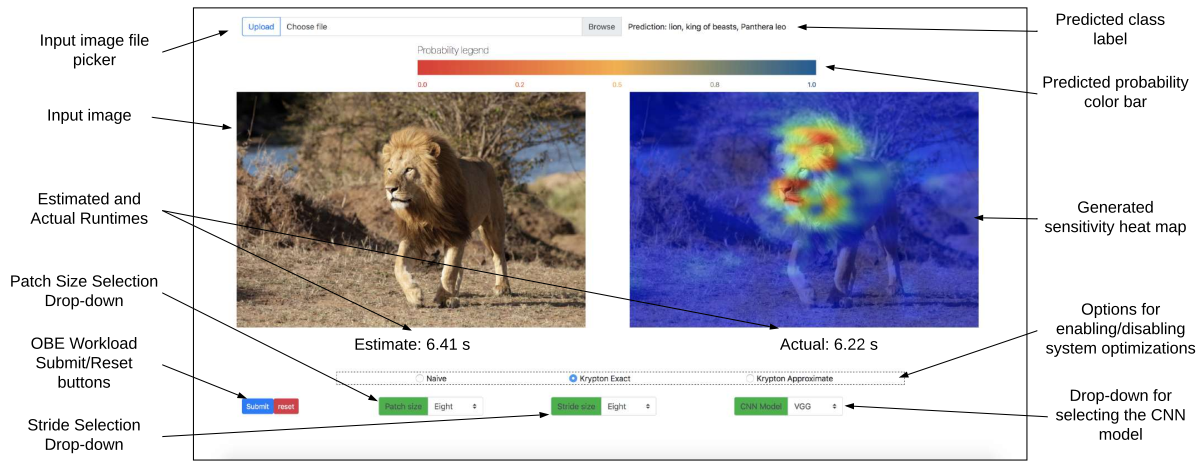
Deep Convolution Neural Networks (CNNs) are now the state of the art method for many image prediction tasks. Thus, there is growing interest in adopting deep CNNs in various application domains, including high stakes applications such as healthcare [1, 7]. Despite their successes, a key criticism of CNNs is that their internal workings are unintuitive to non-technical users. Thus, users often seek an "explanation" for why a CNN predicted a certain label. Explanations can help users trust CNNs [9] and are a legal requirement for machine learning applications in some countries [12]. How to explain a CNN prediction is still an active research question, but in the practical literature, an already popular mechanism for CNN explanations is a simple procedure called *occlusion-based explanations* [13], or OBE for short.

OBE works as follows. Place a small square patch (usually black) on the image to occlude those pixels and rerun CNN inference on the occluded image. The probability of the predicted class will change. Repeat this process by moving the patch across the image to obtain a sensitivity *heatmap* of probability changes, as shown in Figure 1. This heatmap highlights regions of the image that were highly "responsible" for the prediction (red/orange color regions). Such localization of the regions of interest allows users to gain intuition on what "mattered" for the prediction. Overall, OBE is popular because it is easy for non-technical users to understand.

Alas, OBE is highly expensive computationally. Deep CNN inference is already expensive; OBE just amplifies it by issuing a large number of CNN re-inference requests (even 1000s). For example, [14] report 500,000 re-inference requests for 1 image, which took 1hr even on a GPU! Such long wait times can hinder users' ability to consume explanations and reduce their productivity. One could use more compute hardware, if available, since OBE is embarrassingly parallel across re-inference requests. But this may not always be affordable, especially for domain scientists, or feasible in all settings, e.g., in mobile clinical diagnosis. Extra hardware can also raise monetary costs, especially in the cloud.

In this work we use a database-inspired lens to formalize, optimize, and accelerate OBE. We start with a simple but crucial observation: *the occluded images are not disjoint but share most of their pixels; so, most of CNN re-inference computations are redundant.* This observation leads us to connect OBE with two classical data management concerns: *incremental view maintenance* (IVM) [5] and *multi-query optimization* (MQO) [10]. Instead of treating a CNN as a "blackbox," we open it up and formalize *CNN layers* as "queries." Just like how a relational query converts relations to other relations, a CNN layer converts *tensors* (multidimensional arrays) to other tensors. So, we reimagine

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352093


Input image file picker
Input image
Estimated and Actual Runtimes
Patch Size Selection Drop-down
OBE Workload Submit/Reset buttons
Stride Selection Drop-down
Predicted class label
Predicted probability color bar
Generated sensitivity heat map
Options for enabling/disabling system optimizations
Drop-down for selecting the CNN model
Upload
Choose file
Browse
Probability legend
0.0
0.2
0.5
0.8
1.0
Naive
Krypton Exact
Krypton Approximate
Submit
reset
Patch size
Stride size
CNN Model

Prediction: lion, king of beasts, Panthera leo

Estimate: 6.41 s

Actual: 6.22 s

Patch size: Eight

Stride size: Eight

CNN Model: VGG

Figure 1: Krypton user interface. Users can load an input image, select a CNN model, and interactively diagnose the prediction by occluding parts of the full image or part of the image using the cropping tool. Krypton generates a sensitivity heatmap (right image) and iteratively refines it as the user progresses. NB: This figure is best viewed in color, as is standard in the visual computing literature.

OBE as *a set of tensor transformation queries* with incrementally updated inputs. With this fresh database-inspired view, we introduce several *novel CNN-specific query optimization techniques* to accelerate OBE.

We prototype our ideas in the popular deep learning framework PyTorch to create a tool we call Krypton. It works on both CPU and GPU and currently supports a few popular deep CNNs (VGG16, ResNet18, and InceptionV3). Krypton yields up to 35x speedups over the current dominant practice of running re-inference with just batching for producing high-quality approximate heatmaps and up to 5x speedups for producing exact heatmaps. In this demonstration, we allow the audience to use Krypton and interactively diagnose CNN predictions from three real-world image datasets from recent radiology and computer vision literature. A short video of our demonstration can be found here: `https://youtu.be/1OWddbd4n6Y`

2. TECHNICAL CONTRIBUTIONS

The novelty of our system comes from the optimization techniques that it uses for accelerating the OBE workload. In this section, we briefly explain our *incremental CNN inference* and *approximate inference* optimizations. More details on Krypton's optimizations can be found in our technical report [2]. In addition to the above optimizations, in this demonstration we showcase Krypton as an end-to-end system for interactive diagnosis of CNN predictions. Users can now select a subset of the image to run OBE using a cropping tool to exploit their intuitions about what regions might be more important. We also provide runtime estimations for the OBE workload.

2.1 Incremental Inference

For the incremental CNN inference optimization, we *materialize* all tensors produced by the CNN's layers on the given image. For every re-inference request in OBE, instead of rerunning CNN inference from scratch, we treat it as an incremental view maintenance (IVM) query [5], with the "views" being the tensors. We rewrite such queries to

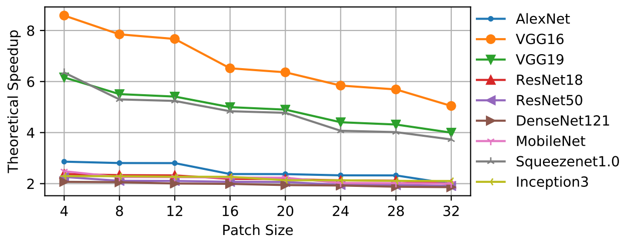


Figure 2: Theoretical speedups for popular deep CNN architectures with incremental inference.

reuse as much of the materialized views as possible and recompute only what is needed, thus *avoiding computational redundancy*. Such rewrites are non-trivial because they are closely tied to the complex geometric dataflows of CNN layers. We have formalized such dataflows to create an *algebraic framework* of CNN query rewrites. Going further, we batch all re-inference requests in OBE to reuse the *same* materialized views. This is a form of multi-query optimization (MQO) [10], albeit interwoven with our IVM, leading to a novel *batched incremental CNN inference* procedure. To the best of our knowledge, this is the first instance of IVM being fused with MQO in query optimization, at least for CNN inference.

We calculate the highest attainable theoretical speedup in terms of the amount of computations saved for performing IVM-based incremental inference for popular CNN architectures with different occlusion patch sizes. The results are shown in Figure 2. VGG-16 has the highest theoretical speedups, while DenseNet-121 has the lowest. Most CNNs fall in the 2x-3x range. The differences arise due to the specifics of the CNN's architecture: VGG-16 has small convolution filter kernels and strides, which means full inference incurs a high computational cost (15 GFLOPs). In turn, incremental inference is most beneficial for VGG-16. Note that we assumed an image size of 224×224 for this plot; if the image is larger, the theoretical speedups will be higher.

While one might be tempted to think that speedups of 2x-3x may not be "that significant" in practice, we find that they indeed are significant for at least two reasons. First, users often wait in the loop for OBE workloads for performing interactive diagnoses and analyses. Thus, even such speedups can improve their productivity, e.g., reducing the time taken on a CPU from about 6min to just 2min, or on a GPU from 1min to just 20s. Second, and equally importantly, incremental inference is the foundation for our approximate inference optimizations, which amplify the speedups we achieve for OBE.

2.2 Approximate Inference

The *approximate inference* optimization in KRYPTON allows users to tolerate some degradation in visual quality of the heatmaps produced to reduce runtimes further in a tunable manner. We draw from the concept of *projective field* from neuroscience and exploit the internal semantics of how CNNs work to reduce runtimes. The projective field of a CNN neuron is the slice of the output tensor that is connected to it. This notion essentially captures the *growth of the size* of the modified patches through the layers of a CNN. Due to the overlapping nature of how convolution filter kernels operate, the projective field of a modified patch in the input image will grow at every layer. It can be shown that in the projective field, the change in the pixels that are radially further away from the center of the patch will be marginal. To exploit this property we introduce the concept of *projective field thresholding*, which essentially truncates the growth of the projective field and saves computations.

2.3 OBE Runtime Estimation

To make the KRYPTON's interface more user friendly, we provide runtime estimations for the OBE workload based on the configurations selected by the user. Showing the estimated runtime for the OBE workload will enable the user to make an informed decision when picking the OBE configurations to trade-off the quality of the heatmap and the available time budget. The runtime of a single OBE workload depends on the width and height of the selected image region, the stride value, occlusion patch size, selected CNN model, and the execution mode (i.e. exact vs approximate). When we control for the occlusion patch size, CNN model, and the execution mode, the runtime of the workload is directly proportional to the number of different occlusion patch positions. We ran several offline experiments with different configurations and recorded the runtimes. These runtimes are then used to fit a linear regression cost model for each patch size, CNN model, and execution mode combination and are then used to predict the runtime for new configuration instances. Figure 3 shows the cost models generated for ResNet18 and VGG16 for a patch size of 16 with the exact execution mode.

3. DEMONSTRATION

3.1 Datasets and CNN models

We will present a demonstration of KRYPTON with three real-world image datasets: 1) identifying diabetic retinopathy from retinal images, 2) identifying pneumonia from chest X-ray images, and 3) identifying objects from natural images in the ImageNet dataset. KRYPTON currently supports three popular CNN architectures: VGG16, ResNet18, and Inception3. Altogether, the audience will be able to interact with our system on nine different settings.

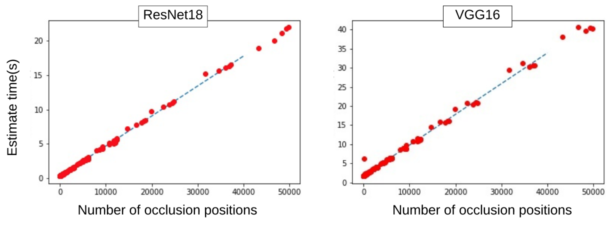


Figure 3: Runtime estimation using linear regression cost model (occlusion patch size = 16 and execution mode is exact).

3.2 Walkthrough

Each participant will be first made familiar with CNN models and OBE method using a supporting slide deck. We will briefly cover important aspects such as different operators inside a CNN, the dataflow inside a CNN, and also the OBE approach for explaining CNN predictions. This brief introduction will give the participants the necessary background to understand the KRYPTON system and appreciate its optimizations. After the introduction, participants will be demonstrated four scenarios of using KRYPTON.

Scenario: Naive OBE. In the first scenario, we will demonstrate performing OBE using the naive approach of performing CNN re-inference for each occlusion patch position. Let's say we want to demonstrate the use of OBE to explain a prediction of an image from the ImageNet dataset from VGG16 model using a patch of size 8×8 and a stride of 2. We first load an image by clicking the file picker shown in Figure 1. After picking an image file, the image will be then displayed on the left hand side panel of the interface. Next we select the Naive option by clicking the corresponding radio button. Selecting of the patch size, stride, and the CNN model can be done using the corresponding drop-down menus as shown in Figure 1. We allow the user to pick from 4 different patch sizes (4×4, 8×8, 16×16, 32×32) and 4 different stride values (2, 4, 8, 16). Currently we support three different CNN model (VGG16, ResNet18, Inception3).

After picking the above configuration values, the user can then initiate the OBE workload by clicking the "Submit" button. The system will also display an estimate for the runtime of the workload. After the workload is completed, the system will overlay the sensitivity heatmap on the original image and display it on the right hand side panel. We also show the predicted class label (e.g., lion) and the actual workload execution time. On the heatmap, red color regions correspond to low predicted probabilities for the selected label (e.g., lion) and the blue color regions correspond to high predicted probabilities for the selected label. So the red color regions are the most sensitive regions for the predicted class label. The color bar for the heatmap is also shown in the interface.

Scenario: KRYPTON Exact. After demonstrating the OBE workload using the naive approach, we then demonstrate the utility of KRYPTON's incremental inference optimizations. We call this KRYPTON Exact execution. Recall that incremental inference optimizations produce the exact same heatmap produced by the naive approach but at a reduced computational cost. Krypton Exact scenario is same as the previous scenario but with the exception of picking the Krypton Exact option from the radio button options instead of Naive option. The estimated and actual runtimes for Krypton Exact will be much smaller than Naive OBE.

Scenario: Krypton Approximate. Next we will demonstrate the utility of Krypton's approximate inference optimizations, which we call Krypton Approximate execution. Approximate inference optimization trades off the accuracy of the generated heatmap with respect to the original heatmap in favor of faster runtimes. Krypton Approximate scenario is also same as the previous scenarios but with the exception of picking the Krypton Approximate option. The estimated and actual runtimes for Krypton Approximate will be even smaller than Krypton Exact. Though there will be minor differences between the generated heatmaps with Krypton Exact and Krypton Approximate, participants will be able see that the overall visual perception of both methods are very much the same.

Scenario: Interactive Image Cropping. In all of the above three cases, we considered occlusion patch positions over the entire image. However, this can be wasteful if the significant objects in an image are localized into a small region. For example consider the image shown in Figure 1. It contains an image of a lion on a wilderness background. Clearly, the main object in this image is the lion which occupies only a small region of the entire image. A user who wants to diagnose the prediction for this image can start the diagnosis by selecting a smaller region which contains only the face and the body of the lion by cropping that region. This will execute faster than the full image. If the user is not satisfied with the produced heatmap, she can iteratively refine the selected region. The cropping of an image can be done simply by dragging on the image in the interface to select a rectangular selection area. Cropping operation is supported with all of the above three scenarios.

After going through the above four scenarios, participants will then be given the opportunity to use the system by themselves. They will be able to select images from three different datasets: OCT images, Chest X-Ray images, and natural images from ImageNet dataset and will have the opportunity to *interactively diagnose CNN predictions*.

4. RELATED WORK

CNN Explanations. Perturbation-based and gradient-based are the two main kinds of CNN explanation methods. Perturbation-based methods observe the output of the CNN by modifying regions of the input image [13]. OBE belongs to this category. Gradient-based methods generate a sensitivity heatmap by computing the partial derivatives of model outputs with respect to every input pixel [11]. However, OBE is usually the method of choice for domain scientific users, especially in radiology [6, 8], since it is easy to understand for non-technical users and typically produces high-quality and well-localized heatmaps. Also to the best of our knowledge, ours is the first work to address the CNN explainability problem from a systems standpoint.

Faster CNN Inference. There are several approaches proposed in the literature for accelerating CNN inference. EVA^2 [3] is a custom software-hardware integrated stack for exploiting temporal redundancy across video frames. Since our optimizations are at the logical level, they are also applicable to any compute hardware. CBinfer performs change-based approximate CNN inference to accelerate real-time object recognition on video [4]. Our focus is on accelerating the OBE workload for images, not video streams. Our IVM and approximate inference optimizations exploit specific semantic properties of OBE, not general object recognition. Overall, both of these tools are orthogonal to our focus.

Acknowledgments. This work was supported by a Hellman Fellowship and by the NIDDK of the NIH under award number R01DK114945. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. We thank NVIDIA Corporation for the donation of the Titan Xp GPU used for this work.

5. REFERENCES

[1] Ai device for detecting diabetic retinopathy earns swift fda approval. `https://www.aao.org/headline/first-ai-screen-diabetic-retinopathy-approved-by-f`. Accessed March 15, 2019.

[2] Incremental and approximate inference for faster occlusion-based deep cnn explanations. `https://adalabucsd.github.io/papers/TR_2019_Krypton.pdf`. Accessed March 15, 2019.

[3] M. Buckler et al. Eva^2: Exploiting temporal redundancy in live computer vision. *arXiv preprint arXiv:1803.06312*, 2018.

[4] L. Cavigelli et al. Cbinfer: Change-based inference for convolutional neural networks on video data. In *11th International Conference on Distributed Smart Cameras*, pages 1–8. ACM, 2017.

[5] R. Chirkova, J. Yang, et al. Materialized views. *Foundations and Trends in Databases*, 4(4):295–405, 2012.

[6] K.-H. Jung et al. Deep learning for medical image analysis: Applications to computed tomography and magnetic resonance imaging. *Hanyang Medical Reviews*, 37(2):61–70, 2017.

[7] D. S. Kermany et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[8] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint arXiv:1706.07269*, 2017.

[9] M. T. Ribeiro et al. Why should i trust you?: Explaining the predictions of any classifier. In *22nd ACM SIGKDD*, pages 1135–1144. ACM, 2016.

[10] T. K. Sellis. Multiple-query optimization. *ACM TODS*, 13(1):23–52, 1988.

[11] K. Simonyan et al. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[12] P. Voigt and A. Von dem Bussche. *The EU General Data Protection Regulation*, volume 18. Springer, 2017.

[13] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.

[14] L. M. Zintgraf et al. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*, 2017.