Revision Response Letter

We thank the reviewers for their feedback and suggestions. We have made the utmost effort to carefully incorporate all of the feedback. We think the paper has improved as a result of these changes. All changes made to the paper have been highlighted with red font color. In this letter, we discuss the changes made and respond to the specific revision items listed in the reviews and metareview.

As an overall summary, we made following major changes to the paper:

1. Added a related work section on CNN explanation methods to **Section 6**. We also evaluated *Axiomatic Attribution for Deep Networks* method against OBE. Due to space constraints we added the runtimes plot and the heatmap visualizations into the **Appendix H**, which we refer from **Section 6**.
2. Added a related work section on methods for accelerating CNN inference to **Section 6**. We referenced EVA^2 method and explained how the optimizations proposed by our system are complementary to EVA^2.
3. Added an experiment to the experiments section (**Section 5.2**) to evaluate the memory overhead associated with incremental inference approach. Due to space constraints the results to the **Appendix F**.
4. We elaborated more on how our system is integrated into PyTorch in the **Appendix B** section and referenced that information from the Experiments section (**Section 5**).
5. We also added details on the broad applicability of our optimizations for other use cases into the Summary and Discussion section (**Section 5.3**).

1. REVIEWER 1

W1: *Writing is very difficult to follow. In particular, the calculation of receptive field across layers (and its efficiency of calculation) is very hard to follow. I am not sure if there is an alternative notation that would be clearer.*

Response:
We have refined the text in receptive field (**Section 3.2**) and efficiency calculation (**Section 3.1**) sections and refer to the corresponding visual depictions when explaining the equations to make the intuition behind the equations more clearer. We believe that having these formal definitions is important as simplifying them would create important unanswered questions.

W2: *Not clear to me OBE is the best choice and other interpretability methods are generally cheaper.*

Response:
From the literature we found that OBE is usually preferred over other methods by domain users, such as in radiology, as they produce high quality fine-grained heat maps using a process very intuitive to the human user [10, 34]. There are more recent versions of OBE such as "Prediction Difference Analysis - ICLR 2017' which underscores the importance of optimizing OBE. In **Section 6** we summarized different approaches that are available for explaining CNN predictions and cited related work that shows the importance of OBE as an explainability approach.

D1: *Very little work in the interpretability space is cite. Most relevant is "Axiomatic Attribution for Deep Networks" which computes per-pixel attributions by comparing importance relative to a black pixel, similar to occlusion, but does so through mathematical analysis of the network, not many inferences. Comparing to this seems important to justify why OBE is better.*

Response:
We evaluated "Axiomatic Attribution of Deep Networks" (IGD) method against OBE and found that OBE can generate more fine-grained heat maps compared to IGD. The runtime of IGD method is dependant on the number of gradient steps used for approximating the integration, which is tunable by the user. The runtime of our system also depends on tunable parameters which CNN specific and dataset specific. From our experiments we found that runtimes of IGD and OBE method with our optimization are comparable to each other. Due to space constraints we included this information in **Appendix H**.

Also see our response to Reviewer 1, W2.

D2: *I believe the components in this systems are individually interesting outside of interpretability. It would be interesting to understand if this ideas would be useful in other inference settings where inputs change only slightly between queries (for example, subsequent frames in a video).*

Response:
In the Summary and Discussion section (**Section 5.3**) we summarized different avenues on which our work can be extended. For this first paper on this work, we decided to use OBE as the concrete use case to demonstrate the effectiveness of our incremental CNN inference framework. Applying our optimizations to other use cases such as video analytics with infrequently changing frames require solving several

open problems such as batching update patches with different sizes to utilize the maximum hardware throughput. We are currently exploring the wide applicability of our framework for such use cases.

2. REVIEWER 2

W1: *The problem of occlusion-based explanations seems rather narrow, and it is not clear whether a more general tool could be used to get similar benefits here. For example, you could represent the occluding box moving around the image as a video, and then use techniques for fast inference on videos, for example the ones in the paper EVA: Exploiting Temporal Redundancy in Live Computer Vision (Buckler et al, ISCA 2018). Comparing against a more general tool would make this paper stronger.*

Response:
We added a related work section on different methods for faster CNN inference (**Section 6**) and summarized various other approaches including EVA^2 and explained how our system is either complementary or orthogonal to those systems.

Focus of EVA^2 system is on exploiting the temporal redundancy of the video frames for faster inference. It does not exploit the spatial redundancy of the images. As the reviewer suggested the different occlusion positions can be treated as a video. However, EVA^2 will still perform motion estimation computations for the entire frame. Optimizations proposed in Krypton exploits the spatial redundancy of frames. Therefore, our work is complementary to EVA^2. Furthermore, our optimizations are logical optimizations. They can be implemented on any hardware platform that support convolution operations.

W2: *Another weakness is that it seems likely that the techniques in Krypton can produce improvements for applications beyond the scope of occlusion-based explanations. For example, how would Krypton perform on an infrequently changing video feed, or one that changes in only a small part of the image? I think a comparison like this would improve the paper.*

Response:
Please see our response to Reviewer 1 D2.

W3: *A third potential weakness lies in the approximate inference section, because there are many ways of doing approximate inference (e.g. low-precision computation, pruning, etc.) and it's not clear whether these could perform better than the new approximate inference methods proposed in this work. It would be an improvement to see some comparison to other methods for accelerating approximate inference.*

Response:
To be more precise on the scope of our optimizations we have refined the introduction paragraph in the Approximate Inference section (**Section 4**). The optimizations performed by our system are logical optimizations and they are complementary to other physical optimizations such as low-precision computation. The optimizations introduced in Krypton are applicable to a pruned version of a CNN as well as it exploits only the spatial redundancy. Thus they are complementary to each other.

D1: *Figure 2 is hard to read. A lot is happening in that diagram for a "simplified illustration" and I think the font should be larger. Figure 15 in the appendix is very difficult to read. You should make the font size larger. In Figure 17 in the appendix, there seems to be an interesting phenomenon in which performance breaks down at a protective field threshold of 0.3 across all three images. This is interesting, and might be worth a sentence or two of discussion.*

Response:
We refined the text in **Figure 2** and **Figure 15** and also increased the font size. We also added a sentence describing quality degradation phenomena to the Appendix **Figure 17** (new **Figure 19**) caption.

3. REVIEWER 5

W1: *The speedups achievable using the technique are dependent on the architectural properties of the CNN. The authors have explicitly identified the limitation in the paper. It would be good if given a CNN architecture, KRYPTON can provide an estimate of the speedup upfront to see if techniques in the paper will be beneficial for the architecture.*

Response:
We evaluate the theoretical bounds for attainable speedup by estimating FLOPs savings for CNN operations (**Section 2.2** Computational Cost of Inference). We also estimate the speedup attainable from adaptive drill-down (**Section 4.2** Theoretical Speedups). These speedups can be used to characterize the benefits based on architecture. Calculating the total cost of inference is largely impractical as it requires calculating the computational cost, memory copy overheads etc.

W2: *Increasingly, CNNs are synthesized by learn to learn techniques. As future work, the authors should consider how a limited projective field can be included as a first class evaluation metric in such synthesis process and if this leads to new architectures that are IVM friendly.*

Response:
We identified this as one of the potential avenues to extend our work in the Summary and Discussion section (**Section 5.3**). We agree with the reviewer and it is something we are currently looking into as an extension of our work.

W3: *The experimental evaluation is largely focused on the speed up obtained when dealing with one image (and its distortions) at a time. Maintaining the output tensors in memory incurs additional memory overhead. This can become significant in a shared serving environment where multiple inference requests (multiple raw images) are processed by the model concurrently. It would be good to have some experiment that also illustrates the memory overhead.*

Response:
We added an additional experiment to evaluate the memory usage behavior to the **Appendix F** and referred it from the main experiments section (**Section 5**). The memory overhead of IVM approach will be much smaller (even up to 52%) than the full inference. Krypton materializes a single copy of all CNN layers corresponding to the unmodified image and reuses it across a batch of occluded images with IVM. For IVM the size of required memory buffers are much smaller than the memory buffers required for full inference as only the updated patches need to be propagated.

D2: *Font size of text in Figure 2 is a bit small.*

Response:
We have revised the text in **Figure 2** and have increased the font size.

D3: *I wonder if such IVM techniques can also be applied to non-CNN models that have certain structural properties. For instance, in case of multi-tower models, the perturbation of input values of one tower do not affect the intermediate tensors of other towers until the final few layers.*

Response:
The redundancies that exists in multi-tower models are very coarse-grained. We can easily exploit these redundancies by simply caching the final layer of the towers that do not change. On the other hand the focus of our system is on more fine-grained redundancies that exists in convolutional layers with small spatially localized updates. However, in a multi-tower CNN model it is possible to integrate both approaches to exploit both coarse-grained and fine-grained redundancies.

4. REVIEWER 6

W1: *Occlusion-based inference is not such a common case, so the use case is small.*

Response:
Please see our response to Reviewer 1 W2.

W2: *Not clear how this generalizes to some of the more complex architectures, such as DenseNet or ResNext – could you please discuss this?*

Response:
We have addressed this in **Section 3.3** "Extending to DAG like CNNs." The complexity of ResNeXt arises from element wise addition operations and the complexity of DenseNet arises from depth wise concatenation operations.

W3: *Could there by other applications in the A/V space beyond partial occlusion that use the same techniques – maybe painting parts of an image, or erasing parts of an image and them repainting it with something new?*

Response:
Please see our response to Reviewer 1 D2.

W4: *There is a section on approximate inference, and it is rather ad-hoc, especially given the plethora of other methods such as quantization, pruning or collapsing a deep net, etc.)*

Response:
Please see our response to Reviewer 2 W3

W5: *Not clear how Krypton is integrated into PyTorch. Is there a special API? What is the architecture of the integration?*

Response:
We have extended information about the integration of Krypton in to PyTorch in **Appendix B** and referred it from the experimental setup section (**Section 5**). Appendix **Figure 15** shows the architecture of the integration. Data is not transferred to the main memory in-between layers.

D1: *I would have liked to see some network architectures where this does not work so well – what are conditions where this is the case? Section 3 discusses this a bit, but could we have a clear characterization based on the architecture – it seems that I can do a pre-calculations that based on the size of the occlusion I can calculate the savings?*

Response:
Yes. Given a CNN model architecture, in **Section 3.1** we provide an approach to estimate the expected speedups by applying our IVM method. Expected speedups is a function of the rate of projective field growth of the CNN which is determined by factors such as number of layers, convolution filter sizes, and filter stride values. For example, as shown in the End-to-End experimental results, VGG16 works really well as it uses small filter kernels and strides. But ResNet18 and Inception3 does not yield gains as high as VGG16.

D2: *It is not clear to me how this is implemented. How do we bring the data back and forth to the GPU? Why are we getting such huge savings – it is really just the number of add/multiplies saved? How do we now organize such irregular computations inside a GPU? Would be great to get answers to these questions.*

Response:
We explain the high level flow of our system in **Section 3.4 Algorithm 1**. Though the updated regions are spatially localized they do not reside in contiguous memory locations. So before we perform the incremental CNN inference we copy the updated regions into a contiguous memory location. Basically, we take a copy of the read-in context from the pre-materialized CNN layer values corresponding to the unmodified image and superimpose the updated patch. We repeat this for all the CNN layers.

Appendix B summarizes how we have implemented this on PyTorch. For the CPU environment we implement this purely on top of PyTorch using built-in tensor slicing and stitching primitives. However, for the GPU environment such iterative memory copying operations introduce high overheads as the many GPU cores now have to idle wait for the slow memory copy operations. To overcome this we extended PyTorch by adding a custom GPU kernel which optimizes the input preparation for incremental inference by invoking parallel memory copy operations. This custom kernel is integrated to PyTorch using Python foreign function interface (FFI). Python FFI integrates with the Custom Kernel Interface layer which then invokes the Custom Memory Copy Kernel Implementation. The high-level architecture of the Custom Kernel integration is shown in **Figure 15**.

The savings mainly come from the saving of redundant computations. However, this introduces memory slicing and stitching overheads which our system tries to minimize using an optimized memory copy kernel. Once the initial input data is transferred to GPU memory, the entire incremental CNN inference can be done using only the GPU memory.

D3: *Are there network architectures where these techniques work especially well or badly?*

Response:
Please see our response to Reviewer 6 D1.

D4: *There is work on self-adjusting computation, where a program learns how to react to changes in its inputs (`http://www.umut-acar.org/self-adjusting-computation`). How does your work compare to this related work?*

Response:
Self-adjusting computation and other reactive computation methods found in the programming languages literature falls into the broader category of general incremental computation approaches. On the other hand the optimizations performed by our systems are specialized optimizations which takes into account the specific dataflow patterns in CNNs, the OBE workload, and also the characteristics of image datasets to perform incremental computations. Therefore, the two lines of work are complementary to each other.

D5: *Section 4.3 has a lot of formulas, but the reader is missing the intuition behind these formulas, Could they be made more accessible?*

Response:
The equations in Section 4.3 are essentially derivations from theoretical speedup calculation explained in **Section 4.2 Equation 28**. To clarify this more we have refined the text in **Section 4.3**.