Kryton: Optimizing Occlusion based Deep CNN Explainability Workloads

Supun Nakandala Arun Kumar
University of California, San Diego
{snakanda,arunkk}@eng.ucsd.edu

ABSTRACT

Deep Convolution Neural Networks (CNN) have revolutionized the field of computer vision with even surpassing human level accuracy in some of the image recognition tasks. Thus they are now being deployed in many real-world use cases ranging from health care, autonomous vehicles, and e-commerce applications. However one of the major criticisms pointed against Deep CNNs is the black-box nature of how they make predictions. This is a critical issue when applying CNN based approaches to critical applications such as in health care where the explainability of the predictions is also very important. For interpreting CNN predictions several approaches have been proposed and one of the widely used method in image classification tasks is occlusion experiments. In occlusion experiments one would mask the regions of the input image using a small gray or black patch and record the change in the predicted label probability. By systematically changing the position of the patch location, a sensitivity map can be generated from which the regions in the input image which influence the predicted class label most can be identified. However, this method requires performing multiple forward passes of CNN inference for explaining a single prediction and hence is very time consuming. We present Krypton, the first data system to elevate occlusion experiments to a declarative level and enable database inspired automated *incremental* and *approximate* inference optimizations. Experiments with real-world datasets and deep CNNs show that Krypton can enable up to 10x speedups.

ACM Reference Format:
Supun Nakandala Arun Kumar. 2018. Kryton: Optimizing Occlusion based Deep CNN Explainability Workloads. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep convolutional neural networks (CNNs) [1, 2] have revolutionized the computer vision field with even resulting near-human accuracies for some image recognition challenges. Many of these successful pre-trained CNNs from computer vision challenges have been successfully repurposed to be used in other real-world image recognition tasks, using a paradigm called *transfer learning* [3]. In transfer learning, instead of training a CNN from scratch, one uses a pre-trained Deep CNN, e.g., ImageNet trained VGG, and fine tune it for the target problem using the target training dataset. This approach avoids the need for a large training datasets, computational power and time which is otherwise a bottleneck for training a CNN from scratch. As a result, this paradigm has enabled the wide adoption of deep CNN technology in variety of real world image recognition tasks in several domains including health care [4, 5], agriculture [6], security [7], and sociology [8]. Remarkably, United States Food and Drug Administration Agency has already approved the use of deep CNN based technologies for identifying diabetic retinopathy, an eye disease found in adults with diabetes. It is expected that above type of decision support systems will help the human radiologists in fulfilling their workloads efficiently and also provide remedy to the shortage of qualified radiologists globally [9].

However, one of the major criticisms for deep CNNs, and deep neural networks in general, is the black-box nature of how they make predictions. In order to apply deep CNN based techniques in critical applications such as health care, the decisions should be explainable so that the practitioners can use their human judgment to decide whether to rely on those predictions or not [10].

In order to improve the explainability of deep CNN predictions several approaches have been proposed. One of the most widely used approach in image recognition tasks is occlusion experiments [11]. In occlusion experiments, a square patch, usually of gray or black color, is used to occlude parts of the image and record the change in the predicted label probability. By systematically striding this patch horizontally and vertically one or two pixels at a time over the image, a sensitivity heatmap for the predicted label can be generated. Using this heatmap, the regions in the image which are highly sensitive (or highly contributing) to the predicted class can be identified. This localization of highly sensitive regions then enables the practitioners to get an idea on the the prediction process of the deep CNN (see Figure. ??).

Example: Consider a radiologist who is examining Optical Coherence Tomography (OCT) images of the retina to identify potential diabetic retinopathy patients. The radiologist is recently given access to a deep CNN based clinical decision support system (CDSS) to identify potential images with diabetic retinopathy. It predicts the probability whether a retina image depicts a diabetic retinopathy case. She uses the CDSS for two main purposes: 1) as a cross checker while manually inspecting the retinal images, and 2) to prioritize potentially sever cases from a backlog of retina images. In both situations in addition to predicting the existence of the disease, the radiologist would like to have an explanation for

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

the basis on which the CDSS makes the prediction, using the occlusion based explainability approach, to decide whether the pathological regions identified by the CDSS are correct and to ultimately whether to rely on the CDSS decision. Similar examples arise in number of other heal care applications such as chest X-ray examination for identifying pneumonia cases and X-ray based child bone age assessment.

However, occlusion experiments are highly compute intensive and time consuming as each patch super imposition is treated as a new image and requires a separate CNN inference. In this work our goal is to apply database inspired optimizations to the occlusion based explainability workload to reduce both the computational cost and runtime taken for an experiment. This will also make occlusion experiments more amenable for interactive diagnosis of CNN predictions. Our main motivation is based on the observation that when performing CNN inference corresponding to each individual patch position, there are lot of redundant computations which can be avoided. To avoid redundant computations we introduce the notion of *incremental inference* of deep CNNs which is inspired by the incremental view maintenance approach which is heavily studied in the context of relational databases.

Due to the overlapping nature of how a convolution kernel would operate (details to follow in Section 3), the size of the modified patch will start growing as it progress through more layers in a CNN and reduce the amount of redundant computations. However at deeper layers the effect over patch coordinates which are radially further away from the center of the patch position will be diminishing. Our second optimization is based on this observation where we apply a form of *approximate inference* which applies a *propagation threshold* to limit the growth of the updating patch. We show that by applying propagation thresholds, a significant amount of computation redundancy can be retained.

The third optimization is also a form of *approximate inference* which we refer as *adaptive drill-down.* In most occlusion experiment use cases, such as in medical imaging, the object or pathological region of interest is contained in a relatively small region of the image. In such situations it is unnecessary to inspect the original image at the same high resolution of striding the occluding patch one or two pixels at a time, at all image locations. In adaptive drill-down approach, first a low resolution heatmap is generated with a larger occluding patch and a larger stride with relatively low computational cost and only the interested regions will be inspected further with a smaller occluding patch and a smaller stride to produce a higher resolution output. This two stage process also reduces the runtime of occlusion experiments.

Unlike the *incremental inference* approach which is exact, *propagation thresholding* and *adaptive drill-down* are approximate approaches. They essentially trade-off accuracy of the generated sensitivity heat map compared to the original, in favor of faster runtime. These changes in accuracy in the generated heat map will be visible all the way from quality differences which are almost indistinguishable to the human eye to drastic structural differences, depending on the level of approximation. This opens up an interesting trade-off space of quality/accuracy versus runtime. Krypton expects the user to define the required level of quality for the generated heat maps by specifying the Structural Similarity Index (SSIM) (explained in Section 3) quality metric. The system will then automatically tune it's *approximate inference* configuration values to yield the expected quality level during an initial tuning phase.

Finally, we have implemented Krypton on top of PyTorch deep learning toolkit by extending it by adding custom implementations of incremental and approximate inference operators. It currently supports VGG16, ResNet18, and InceptionV3 both on CPU and GPU environments, which are three widely used deep CNN architectures for transfer learning applications. We evaluate our system on three real-world datasets, 1) diabetic retinopathy (DR), 2) chest X-ray (CX), and 3) bone age X-Ray (BX), and show that Krypton can result in up to 10x speedups with hardly distinguishable quality differences in the generated occlusion heat maps. While we have implemented Krypton on top of PyTorch toolkit, our work is largely orthogonal to choice of the deep learning toolkit; one could replace PyTorch with TensorFlow, Caffe2, CNTK, MXNet, or implement from scratch using C/CUDA and still benefit from our optimizations. Overall, this paper makes the following contributions:

- To the best of our knowledge, this is the first paper to study

Outline. The rest of this paper is organized as follows.

2 BACKGROUND

Deep CNNs. Deep CNNs are a type of neural networks specialized for image data. They exploit spatial locality of information in image pixels to construct a hierarchy of parametric feature extractors and transformers organized as layers of various types: *convolutions*, which use image filters from graphics, except with variable filter weights, to extract features; *pooling*, which subsamples features in a spatial locality-aware way; *batch-normalization*, which normalizes the output of the layer; *non-linearity*, which applies a non-linear transformation (e.g., ReLU); *fully connected*, which is a multi-layer perceptron; and *softmax*, which emits predicted probabilities to each class label. In most "deep" CNN architectures, above layers are simply stacked together with ones output is simply fed as the input to the other, while adding multiple layers element-wise or stacking multiple layers together to produce a new layer is also present in some architectures. Popular deep CNN model architectures include AlexNet [1], VGG [2], Inception [12], ResNet [13], SqueezeNet [14], and MobileNet [15]. In this work, the discussion and evaluation is focused on VGG-16 (16 layer version), ResNet-18 (18 layer version) and Inception-V3 (version 3) which are three widely used CNN models in real world transfer learning applications. Nevertheless, our work is orthogonal to the specifics of a particular architecture and the proposed approaches can be easily extended to any architecture.

Deep CNN Explainability With image classification models, natural question is if the model is truly identifying objects in the image or just using surrounding or other objects for making false prediction. The various approaches used to explain CNN predictions can be broadly divided into two categories, namely gradient based and perturbation based approaches. Gradient based approaches generate a sensitivity map by computing the partial derivatives of model output with respect to every input pixel via back propagation. In perturbation based approaches the output of the model is observed by masking out regions in the input image and there by identify the sensitive regions. The most popular perturbation based approach is occlusion experiments which was first introduced by Zeiler et. al. [11]. Even though gradient approaches require only a single forward inference and a single backpropagation to generate the sensitivity map, the output may not be very intuitive and hard to understand because the salient pixels tend to spread over a very large are of the input image. Also as explained in [16], the back-propagation based methods are based on the AI researchers intuition of what constitutes a good explanation. But if the focus is on explaining decision to a human observer, then the approach used to produce the explanation should have a structure that humans accept. As a result in most real world use cases such as in medical imaging, practitioners tend to use occlusion experiments as the preferred approach for explanations despite being time consuming, as they produce high quality fine grained sensitivity maps [10].

Over the years there has been several modifications proposed to the original occlusion experiment approach. More recently Zintgraf. et. al. [17] proposed a variation to the original occlusion experiment approach named *Prediction Difference Analysis*. In their method instead of masking with a gray or black patch, samples from surrounding regions in the image are chosen as occlusion patches. In our work we mainly focus on the original occlusion experiment method. But, the methods and optimizations proposed in our work are readily applicable to more advanced occlusion based explainability approaches.

3 PRELIMINARIES AND OVERVIEW

In this section we formalize the internals of some of the layers in a Deep CNN for the purpose of proposing our incremental inference approach in Section 4. Next we briefly explain the Structural Similarity Index (SSIM) which is used to quantify the quality of the generated sensitivity heat map with respect to the original in the context of approximate inference. Finally we formally state the problem, explain our assumptions.

3.1 Deep CNN Internals

The output activations of the layers in a Deep CNN, except for fully-connected ones, are arranged into three dimensional volumes which has a width, height, and depth. For example an RGB input image of 224×224 spatial size can be considered as an input volume having a width and height of 224 and a depth of 3 (corresponding to 3 color channels). Every non fully-connected layer will take in an input activation volume and transform it into another activation volume, where as a fully-connected layer will transform an input volume into an output vector. For our purpose these transformations can be broadly divided into three subcategories based on how they spatially operate:

- Transformations that operate on individual spatial locations.
  - E.g. ReLU, Batch Normalization
- Transformations that operate on a local spatial context.
  - E.g. Convolution, Pooling
- Transformation that operate on a global spatial context.
  - E.g. Fully-Connected

With incremental spatial updates in the input, both types of transformations that operate at individual spatial locations and transformations that operate at a local spatial contexts provide opportunities for exploiting redundancy. Extending the transformations that operate at individual spatial locations to become redundancy aware is straightforward. However, with transformations that operate on a local spatial context such as convolution and pooling, this extension is non-trivial due to the overlapping nature of the spatial contexts corresponding to individual transformations. We next formally define the transformations of convolution and pooling layers and also the relationship between input and output dimensions for these layers which will be later used in Section. 4 to introduce our incremental inference approach.

Transformations that operate on a local spatial context. The two types of transformations that operate on a local spatial context in a deep CNN are convolution and max pooling layers. Convolution layers are the most important type of layer in a CNN architecture that also contributes to most of the computational cost. Each convolutional layer can have several (say C_{out}) three dimensional filter kernels organized into a four dimensional array $\mathcal{K}_{conv}$ with each having a smaller spatial width W_k and height H_k compared to the width W_{in} and height H_{in} of the input volume $\mathcal{I}$, but has the same depth C_{in}. During inference, each filter kernel is slided along the width and height dimensions of the input and a two dimensional activation map is produced by taking element-wise product between the kernel and the input and adding a bias value $\mathcal{B}[c]$ for some c $\in [0, C_{out} - 1]$. These two dimensional activation maps are then stacked together along the depth dimension to produce an output volume O having the dimensions of (C_{out},H_{out},W_{out}). This can be formally defined as follows:

$$\text{Input Volume} : \mathcal{I} \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}} \quad (1)$$

$$\text{Conv. Filters} : \mathcal{K}_{conv} \in \mathbb{R}^{C_{out} \times C_{in} \times H_k \times W_k} \quad (2)$$

$$\text{Conv. Bias Vector} : \mathcal{B}_{conv} \in \mathbb{R}^{C_{out}} \quad (3)$$

$$\text{Output Volume} : O \in \mathbb{R}^{C_{out} \times H_{out} \times W_{out}} \quad (4)$$

$$\begin{aligned} O[c, y, x] = \sum_{k=0}^{C_{in}} \sum_{j=0}^{H_k-1} \sum_{i=0}^{W_k-1} \mathcal{K}_{conv}[c, k, j, i] \\ \times \mathcal{I}[k, y - \lfloor \frac{H_k}{2} \rfloor + j, x - \lfloor \frac{W_k}{2} \rfloor + i] + \mathcal{B}[c] \end{aligned} \quad (5)$$

The main objective of having pooling layers in CNNs is to reduce the spatial size of output volumes. Pooling can also be thought as a convolution operation with a fixed (i.e. not learned) two dimensional filter kernel $\mathcal{K}_{pool}$ having a width of W_k and height of H_k, which unlike convolution, operates independently on every depth slice of the input volume. The two main variations of pooling layers are max pooling (takes the maximum value from the local spatial context) and average (takes the average value from the local spatial context) pooling. A Pooling layer takes a three dimensional activation volume O having a depth of C, width of W_{in}, and height of H_{in} as input and produces another three dimensional activation volume O which has the same depth of C, width of W_{out}, and height of H_{out} as the output. Pooling kernel is generally slided with more than one pixel at a time and hence W_{out} and H_{out} are

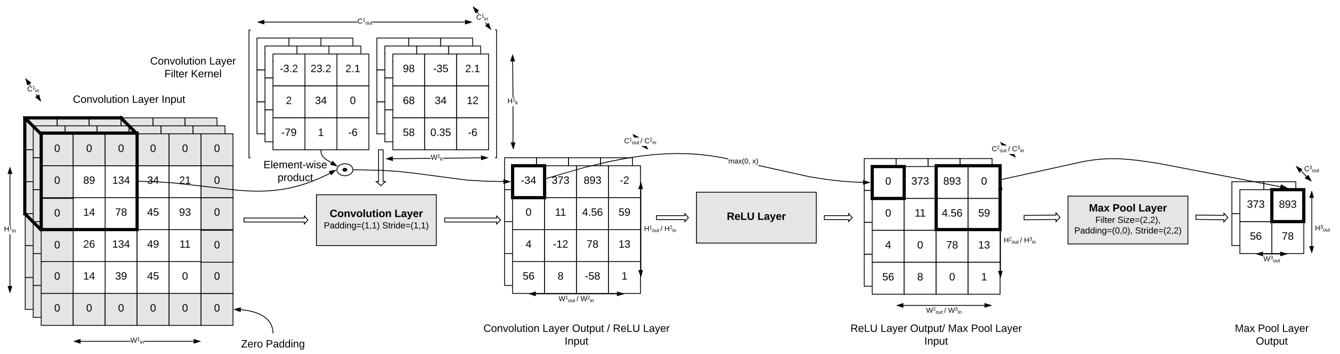

Figure 1: Simplified representation of selected layers of a Deep CNN.

generally smaller than W_{in} and H_{in}. Pooling operation can be formally defined as follows:

$$\text{Pool Filters} : \mathcal{K}_{pool} \in \mathbb{R}^{H_k \times W_k} \tag{6}$$

$$O[c,y,x] = \sum_{j=0}^{H_k-1} \sum_{i=0}^{W_k-1} \mathcal{K}_{pool}[j,i] \times \mathcal{I}[c, y - \lfloor \frac{H_k}{2} \rfloor + j, x - \lfloor \frac{W_k}{2} \rfloor + i] \tag{7}$$

Relationship between Input and Output Spatial Sizes. The output volume's spatial size (W_{out} and H_{out}) is determined by the spatial size of the input volume (W_{out} and H_{out}), spatial size of the filter kernel (W_k and H_k) and two other parameters: **stride** S and **padding** P. Stride is the amount of pixel values used to slide the filter kernel at a time when producing a two dimensional activation map. It is possible to have two different values with one for the width dimension (S_x) and one for the height dimension (H_x). Sometimes in order to control the spatial size of the output activation map to be same as the input activation map, one needs to pad the input feature map with zeros around the spatial border. Padding (P) captures the amount of zeros that needs to be added. Similar to the stride S, it is possible to have two separate values for padding with one for the width dimension P_x and one for the height dimension P_y. With these parameters defined the width and the height of the output activation volume can be defined as follows:

$$W_{out} = (W_{in} - W_k + 2 \times P_x)/S_x + 1 \tag{8}$$
$$H_{out} = (H_{in} - H_k + 2 \times P_y)/S_y + 1 \tag{9}$$

Estimating the Computational Cost of Deep CNNs Deep CNNs are highly compute intensive and out of the different types of layers, Conv layers contributes to 90% (or more) of the computations. One of the widely used way to estimate the computational cost of a Deep CNN is to estimate the number of fused multiply add (FMA) floating point operations (FLOPs) required by convolution layers for a single forward inference and ignore the computational cost of other layers (e.g. Pooling, Fully-Connected).

For example, applying a convolution filter having the dimensions of (C^l_{in}, H^l_k, W^l_k) to a single spatial context will require $C^l_{in} \times H^l_k \times W^l_k$ many FLOPs, each corresponding to a single element-wise multiplication. Thus, the total amount of computations Q_l required by that layer in order to produce an output O having dimensions $C^l_{out} \times H^l_{out} \times W^l_{out}$, and the total amount of computations Q required to process the entire set of convolution layers L in the CNN can be calculated as per equation. 10 and equation. 11. However, in the case incremental updates effectively only a smaller spatial patch having a width W^l_p (W^l_p <= W^l_{out}) and height H^l_p (H^l_p <= H^l_{out}) is needed to be recomputed. The amount of computations required for the incremental computation Q^l_{inc} and total amount of incremental computations Q_{inc} required for the entire set of convolution layers L will be smaller than the above full computation values and can be calculated as per equation. 12 and equation. 13.

Based on the above quantities we define a new metric named **redundancy ratio** R, which is the ratio between total full computational cost Q and total incremental computation cost Q_{inc} (see equation. 14). This ratio essentially acts as a surrogate for the theoretical upper-bound for computational and runtime savings that can be achieved by applying incremental computations to deep CNNs.

$$Q^l = (C^l_{in} \times H^l_k \times W^l_k) \times (C^l_{out} \times H^l_{out} \times W^l_{out}) \tag{10}$$
$$Q = \sum_{l \in L} Q^l \tag{11}$$
$$Q^l_{inc} = (C^l_{in} \times H^l_k \times W^l_k) \times (C^l_{out} \times H^l_p \times W^l_p) \tag{12}$$
$$Q_{inc} = \sum_{l \in L} Q^l_{inc} \tag{13}$$
$$R = \frac{Q}{Q_{inc}} \tag{14}$$

3.2 Estimating the Quality of Generated Approximate Heat Maps

When applying approximate inference optimizations, Krypton sacrifices the the accuracy of the generated heat map in favor of lesser runtime. To measure this drop of accuracy we use Structural Similarity (SSIM) Index [18], which is one of the widely used approaches to measure the *human perceived difference* between two images.

When applying SSIM index, we treat the original heat map as the reference image with no distortions and the perceived image similarity of the KRYPTON generated heat map is calculated with reference to it. The generated SSIM index is a value between −1 and 1, where 1 corresponds to perfect similarity. It is important to note that, even though SSIM index value of 1 corresponds to perfect similarity, other values doesn't necessarily imply same level of perceived quality across different image pairs. However, if the original images are closely similar, such as in chest X-ray images, it can be assumed that this condition will hold. Typically SSIM index values above 0.9 are used in practical applications such as image compression and video encoding as they produce very small distortions at the human perception level. For more details on SSIM Index method, we refer the reader to the original SSIM Index paper [18].

3.3 Problem Statement and Assumptions

4 OPTIMIZATIONS

4.1 Incremental Computation

4.2 Approximate Computation

4.2.1 Adaptive Drill-Down.

4.2.2 Patch Propagation Thresholding.

4.3 System Tuning

5 EXPERIMENTAL EVALUATION

5.1 End-to-End Evaluation

5.2 Drill-Down Analysis

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[3] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802, 2016.

[4] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[5] Mohammad Tariqul Islam, Md Abdul Aowal, Ahmed Tahseen Minhaz, and Khalid Ashraf. Abnormality detection and localization in chest x-rays using deep convolutional neural networks. *arXiv preprint arXiv:1705.09850*, 2017.

[6] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[7] Farhad Arbabzadah, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Identifying individual facial expressions by deconstructing a neural network. In *German Conference on Pattern Recognition*, pages 344–354. Springer, 2016.

[8] Yilun Wang and Michal Kosinski. Deep neural networks are more accurate than humans at detecting sexual orientation from facial images. 2017.

[9] Radiologists are often in short supply and overworked deep learning to the rescue. https://government.diginomica.com/2017/12/20/radiologists-often-short-supply-overworked-deep-learning-rescue. Accessed September 31, 2018.

[10] Kyu-Hwan Jung, Hyunho Park, and Woochan Hwang. Deep learning for medical image analysis: Applications to computed tomography and magnetic resonance imaging. *Hanyang Medical Reviews*, 37(2):61–70, 2017.

[11] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

[17] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*, 2017.

[18] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.