Characterizing and Taming Model Instability Across Edge Devices

Eyal Cidon [*] [1] Evgenya Pergament [*] [1] Zain Asgar [1] Asaf Cidon [2] Sachin Katti [1]

Abstract

The same machine learning model running on different edge devices may produce highly-divergent outputs on a nearly-identical input. Possible reasons for the divergence include differences in the device sensors, the device's signal processing hardware and software, and its operating system and processors. This paper presents the first methodical characterization of the variations in model prediction across real-world mobile devices. We demonstrate that accuracy is not a useful metric to characterize prediction divergence, and introduce a new metric, instability, which captures this variation. We characterize different sources for instability, and show that differences in compression formats and image signal processing account for significant instability in object classification models. Notably, in our experiments, 14-17% of images produced divergent classifications across one or more phone models. We evaluate three different techniques for reducing instability. In particular, we adapt prior work on making models robust to noise in order to fine-tune models to be robust to variations across edge devices. We demonstrate our fine-tuning techniques reduce instability by 75%.

1 Introduction

Machine learning (ML) models are increasingly being deployed on a vast array of devices, including a wide variety of computers, servers, phones, cameras and other embedded devices (Wu et al., 2019). The increase in the diversity of edge devices, and their bounded computational and memory capabilities, has led to extensive research on optimizing ML models for real-time inference on the edge. However, prior work primarily focuses on the model's properties rather than on how each device introduces variations to the input. They tend to evaluate the models on public, well-known and hand-labeled datasets. However, these datasets do not necessarily represent the transformations performed by these devices on the input to the model (Recht et al., 2019a; Torralba & Efros, 2011). The wide variety of sensors and hardware, different geographic locations (Rameshwar, 2019) and processing pipelines all effect the input data and the efficacy of the ML model.

To demonstrate how even small real-world variations to the input can affect the output, in Figure 1 we show an example where pictures of the same object taken by the same phone, one second apart, from a fixed position (without moving the phone), produce different classification results on the same model. Such divergence occurs even more frequently when the same model is run on different devices. In order to understand how the model will perform when run on many different devices, it is important to ensure consistent model accuracy, and create representative training and evaluation datasets that will be robust to variations across devices and sensors.

To capture this variability, we introduce the notion of *instability*, which denotes the probability that the same model outputs different contradictory predictions on nearly identical input when run on different edge devices. Running models on a wide array of edge devices can create many different opportunities for instability. Some of the instability is caused by changes to the inputs to the model, for example by the usage of different sensors on different devices (e.g. camera lenses), applying different transformations on the raw captured data (e.g. image processing pipelines), and saving the data in different formats (e.g. compression). Further instability may be introduced by hardware on the device (e.g. GPUs handling of floating points) (Jia & Rinard, 2020) or the operating system (e.g. the OS stores pictures in a particular format).

Prior work on model robustness has either focused on simulating the sources of instability (Buckler et al., 2017), or on making models robust to adversaries (Kurakin et al., 2017; Bastani et al., 2016; Cissé et al., 2017). While adversarial learning is important for model robustness, the vast ma-

[*]Equal contribution [1]The Department of Electrical Engineering, Stanford University, Stanford, California, USA [2]Electrical Engineering Department, Columbia University, New York, New York, USA. Correspondence to: Eyal Cidon <ecidon@stanford.edu>, Evgenya Pergament <evgenyap@stanford.edu >.

Proceedings of the 4^{th} MLSys Conference, San Jose, CA, USA, 2021. Copyright 2021 by the author(s).

jority of instability in many applications is not caused by adversaries, but rather by normal variations in the device's hardware and software. Therefore, we lack a solid understanding of the *sources of instability in real-world edge devices*.

In this work, we conduct the first systematic characterization of the causes and degree of variance in convolutional neural network (CNN) computer vision model efficacy across existing mobile devices. We conduct four sets of experiments, in which we try to isolate different sources of variance, and evaluate the degree of instability they cause. In each experiment below, the same model is used while we vary the input or the operating conditions and the outputs are compared:

1. **End-to-end instability across phones:** We evaluate end-to-end instability across 5 mobile devices in a lab environment, in which the phones' position and lighting conditions are tightly controlled.
2. **Image compression algorithm:** We evaluate the effect of image compression techniques, by compressing a raw image using different algorithms.
3. **Image signal processor (ISP):** We estimate the effect of using different ISPs, by comparing raw converted images using ImageMagick and Adobe Photoshop.
4. **Processor and operating system:** We evaluate the effect of the device's OS and hardware by running inference on the same set of input images across 5 phones.

Our characterization leaves us with several takeaways. First, we demonstrate that accuracy fails to account for the lack of consistency of predictions across different devices, and motivate the need to focus on minimizing instability as an objective function. Second, we show that instability across devices is significant; between 14-17% of images classified by MobileNetV2 have divergent predictions (both correct and incorrect) in at least two phones. Third, we show that a significant source of instability is due to variable image compression and ISPs. Finally, we do not find evidence that the devices' processors or operating system are a significant source of instability.

While the focus of this work is on systematically characterizing the sources of instability, we also provide a preliminary analysis of mitigation techniques. First, we explore whether fine-tuning the training process by augmenting the training dataset with noise or with photos taken from multiple different phone models would make the model more robust. Inspired by prior work on noise robustness (Zheng et al., 2016), we show that augmenting the training data set with such "noise", can reduce the instability by up to about 75%. Second, once a model is trained, instability can be further reduced if the model can either operate on raw images (when feasible) or by allowing the classification to display additional results (e.g. the top three results instead of the top one result). Our future work will focus on a more systematic development of instability mitigation techniques to handle edge variance.

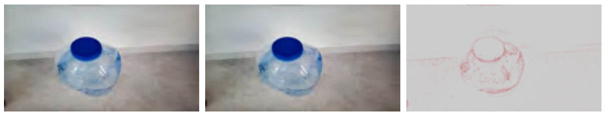


Figure 1. Two photos taken one second apart with Samsung Galaxy S10 in a controlled environment, where the phone was not touched between the two shots. MobileNetV2 assigns the incorrect label "bubble" on the left image and the correct label "water bottle" on the middle image. The right image shows the pixel difference between both photos. We can see the noise is correlated along the edges. There is a very small pixel difference between the images, yet it is sufficient to cause instability.

The paper makes the following contributions:

- **Instability:** We motivate and propose instability, a new metric for evaluating the variability of model predictions across different edge devices.
- **Characterization:** We present the first comprehensive characterization of the end-to-end degree and potential sources of instability on real-world mobile devices.
- **Mitigation:** We propose and evaluate three potential mitigation strategies for instability.
- **Stability Training for Devices:** We adapt a prior approach to make computer vision models robust to random noise (Zheng et al., 2016) for reducing instability across devices.

2 Background

In this section, we provide background for why inference is increasingly being conducted on edge devices, and introduce and define the model instability problem.

2.1 Inference on Edge Devices

While much of model training occurs on servers or datacenters, there is an increasing trend of pushing inference to edge devices (Wu et al., 2019; Lee et al., 2019). The advantages of conducting inference on edge devices over a centralized location are manifold. First, it removes the need to transfer the input data over the network. Second, it can provide lower latency. Third, users may not be always be in settings where they have a stable connection. Fourth, it can ensure the input data collected on the edge remains private.

There are many examples of real-time applications that conduct inference on the edge, including augmenting images taken from a camera (Wu et al., 2019; Ignatov et al., 2017), localizing and classifying objects (Huang et al., 2017), monitoring vital health signals (Miliard, 2019) and recognizing speech for transcription or translation (Schalkwyk, 2019).

However, running inference on different edge devices, each with its own hardware and software, as well as different sensors with which they capture input data, creates very heterogeneous environments. These different environments lead the model to behave differently, even on seemingly identical inputs.

2.2 Model Instability

While the traditional ML metrics, of accuracy, precision and recall are extremely important, they do not capture how well a model performs *across environments*. For this purpose, we define a new metric, called **model instability**, which denotes when a model conducts inference in different environments, and returns significantly different results on near-identical inputs.

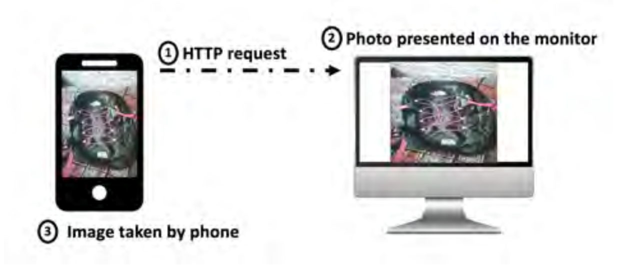
Photography always introduces some random noise during image acquisition from the sensor (Boncelet, 2009). Due to this, models have some degree of instability, even when they are run on exactly the same edge device. To illustrate this, we run the following simple experiment. Figure 1 depicts two photos of a water bottle from the same Samsung Galaxy S10. The photos were taken couple of seconds apart using Android debug bridge, without touching the phone or changing its location. Both images seem identical to the naked eye. However, when we run MobileNetV2 (Sandler et al., 2018) on both models, the one on the left returns the "bubble" class (incorrect), while the model predicts "water bottle" class (correct) for the center image. The image on the right shows the pixels where the difference between in the images is higher than 5%. This experiment shows that even on the same phone, two photos taken within a very short timespan may lead to different predictions. Note the noise across both photos is correlated along the edges.

In order to quantify the measure of instability of a model, we define a prediction as *unstable* if in at least one environment it returns a correct class, and in at least one other environment it returns a clearly incorrect class. We do not compare the predictions between different environments in the case where all the predictions are incorrect, because it is difficult to say whether a particular classification is more "incorrect" than another.

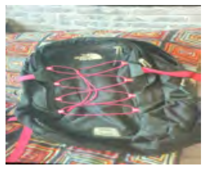
We define the *instability* of a model as the percentage of unstable images out of the total images in our dataset:

$$\frac{\sum_{x \in X} \mathbf{1}[\exists\, i, j \in P \mid (\hat{y}_i = y) \wedge (\hat{y}_j \neq y)]}{\#images}$$

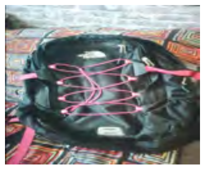
Where X is a set of all images, P is the set of all environments (e.g. phones or compression formats), y is the true label, $\hat{y}_i$ is the prediction of a model on an image x under an environment i, and $\mathbf{1}$ is the indicator function.



(a) end-to-end experiment



(b) iPhone XR



(c) Galaxy S10

Figure 2. (a) Illustration of experimental setup. Each phone sends an HTTP request to a server to display an image. After the image is presented on the monitor, the photo is taken by the phone. (b) Example of an iPhone photo incorrectly classified as "pillow". (c) Example of Samsung Galaxy S10 photo correctly classified as "backpack".

Much of prior work focused on making models robust to adversaries (Kurakin et al., 2017; Bastani et al., 2016; Cissé et al., 2017). In contrast, our focus in this work is on making models robust to naturally occurring variations in model output due to different processors, software and sensors.

3 Experiment Overview

This section details our data collection, experimental setup and the goals of our experiments.

3.1 Data Collection

In order to run our experiments we collected images representing a subset of 5 classes from ImageNet (Deng et al., 2009): water bottle, beer bottle, wine bottle, purse and backpack. The images were a mix of images scraped from Flickr, images downloaded from Amazon and Amazon Prime Now and photos we took ourselves. We collected a total of 1,537 images.

3.2 Experimental Setup

Our goal is to eliminate as much as possible any source of variability that is external to internal characteristics of the phone (e.g. lighting, position, the object being photographed). To this end, we designed the following controlled experimental setup. We placed 5 phones on a camera mount in front of a computer screen in a closed room with light-blocking curtains. The phones took a series of photos of objects presented on the computer screen from the same angle. The objects and photos projected on the screen were taken from our collected dataset described previously (§3.1). Each phone was presented identical photos on the screen. To ensure the phone and computer screen are synchronized, the time at which the photos are taken from the phone is controlled by apps, which we wrote for both iOS and Android. Figure 2 depicts the process: the app communicated with the computer screen, and determines which photo is currently displayed on the screen. The screen displays the photo, and then the app takes a picture of the screen from the device. We repeated this sequence on each of the distinct objects 5 times. In total, we took 68,125 photos.

Table 1. Phones models used in experiments.

PHONE	MODEL
SAMSUNG GALAXY S10	SM-G973U1
LG K10 LTE	K425
HTC DESIRE 10 LIFESTYLE	DESIRE 10
MOTOROLA MOTO G5	XT1670
IPHONE XR	A1984

Throughout the experiments presented in the paper, we evaluate the performance of a single MobileNetV2 (Sandler et al., 2018) model with fixed weights, on the photos taken by the different phones. The model weights were pre-trained on ImageNet. The list of the phones used in our experiments is available in Table 1.

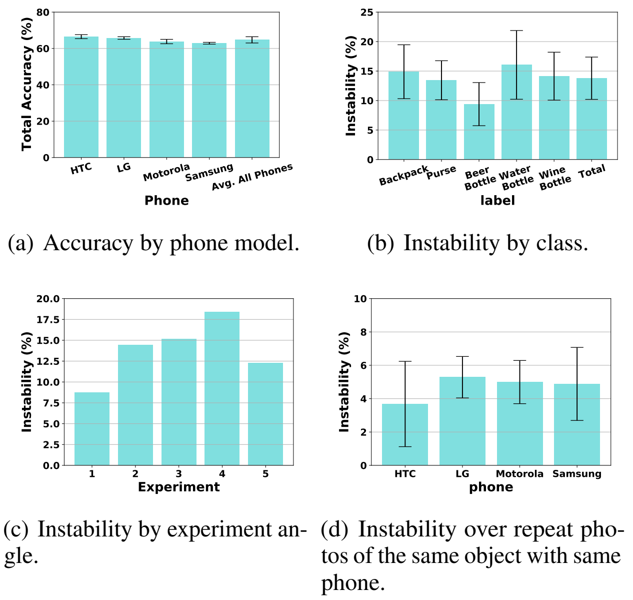
To evaluate the results, we verified whether images were classified correctly or not by hand. For some predictions, there can be more than one possible correct label. For instance, “wine bottle” and “red wine” overlap in ImageNet, so for a bottle of red wine we accepted both ”wine bottle” and ”red wine”. If an image contained more than one object, we only considered the object that is clearly in the foreground of the image.

3.3 Goals

We run four sets of experiments, presented in the next sections: (a) end-to-end experiments, whose goal is to measure the end-to-end instability of models using the same mobile devices and across devices (§4); (b) measuring the effect of image compression (§5); (c) measuring the effect of different ISPs (§6); and (d) measuring the effect of the device’s operating system and processor (§7).

4 END-TO-END INSTABILITY

In our first set of experiments, we seek to quantify the amount of instability across classification tasks within the same phone model and across phone models, and to evaluate



(a) Accuracy by phone model.

(b) Instability by class.

(c) Instability by experiment angle.

(d) Instability over repeat photos of the same object with same phone.

Figure 3. End-to-end accuracy and instability. Accuracy does not capture the variability in predictions across different models.

whether accuracy is a sufficient metric for capturing the variability in classification across edge devices.

4.1 Accuracy vs. Instability

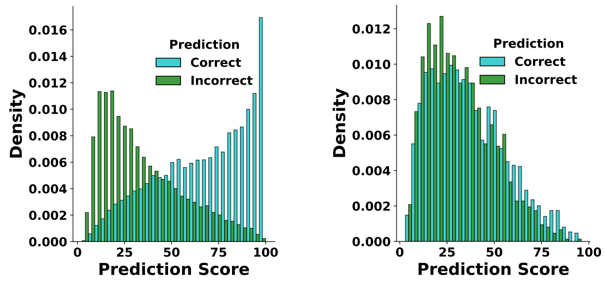
We first evaluated the accuracy of our models on all 5 phones. The results, presented in Figure 3(a), show that accuracy is generally similar across all phone photos, and ranged between $59\% - 64\%$.

The instability across all models is depicted in Figure 3(b). Instability is measured as the percentage of photos, where at least one of the phones was correct (e.g. classified the right class) and at least one of the phones was incorrect, when taking a photo of the same image on the computer screen. The results show that while the accuracy remains relatively stable across the different phones, there is a high degree of instability: for most of the classes about 15% of the images yield at least one correct and one incorrect classification.

The results also demonstrate a large degree of variance in the instability between the different classes; some of them are more prone to instability. Importantly, *this variation is not captured by the accuracy of each class.*

Figure 3(c) plots the variation in instability across the five angles, from left to right, and shows that instability does vary somewhat based on the angle of the image. Figure 3(d) shows that there is instability even across experiments (i.e. different angles) within the same phone model. However, this instability is much lower than the instability across different models.

Based on the results of the experiments, we can conclude accuracy is not a good metric to capture the variability of how well models perform across different devices.



(a) Stable images. (b) Unstable photos.

Figure 4. Prediction score for stable and unstable images.

4.2 Model Confidence

We evaluate the relationship between instability and the model's confidence in its prediction. Figure 4(a) shows the distribution of the prediction scores for stable photos (i.e. those where all phones were correct or all were incorrect). We can see there is a clear correlation between the score and whether the model was correct or not.

Figure 4(b) shows that for unstable predictions (i.e. those where one phone was correct and one was incorrect), the prediction confidence of correct classifications tends to be almost identical to the confidence of the incorrect classifications. This implies that for most unstable images, the model has low confidence whereby even a small amount of noise can cause it to change its prediction. Nevertheless, there is a noticeable group of outlier photos for which the model has very high confidence with one phone but low confidence with the other phone.

5 Image Compression

We now analyze the source and degree of instability within each phone, first focusing on compression. In this section, we analyze only the raw photos taken in the end-to-end experiment on the iPhone and Samsung phone. The phones use two different compression schemes: Samsung uses JPEG and iPhone uses HEIF. In order to isolate the effect of the ISP, we always use ImageMagick to compress and convert the photo to different formats.

5.1 Compression Quality

We compress the photos to JPEG with 3 different qualities: 100, 85 and 50. We used the compression parameter suggested by Google for machine learning (Google, 2019).

Table 2 shows the accuracy using different compression

Table 2. Accuracy and image size for different JPEG compression qualities.

Metric	JPEG 100	JPEG 85	JPEG 50
Avg. Size [MB]	3.05	0.65	0.25
Accuracy	54.0%	54.3%	54.5%
Instability		7.6%	

qualities. The results suggest that using Google's recommended compression parameters leads to very small differences in accuracy across compression qualities. In fact, in our experiment a higher compression ratio surprisingly yielded better accuracy. Yet the instability between the different qualities is 7.6%. Once again, despite the small difference in accuracy across the compression qualities, there is a noticeable instability.

5.2 Different Compression Formats

We repeat this experiment but this time we compress the raw images into different formats: JPEG, PNG, WebP and HEIF. Each format uses its default compression parameters.

Table 3. Image size, accuracy and instability for different compression formats.

Metric	JPEG	PNG	WebP	HEIF
Avg. Size [MB]	1.54	6.49	0.29	0.57
Accuracy	53.9%	53.9%	55.2%	54.4%
Instability		9.66%		

In Table 3 we can see that as before, the different compression format led to small differences in accuracy. Yet the instability across formats is 9.66%. Figure 5 shows a few examples of instability across the different compression formats. Notably the images are almost identical to the naked eye, but produce different model predictions.

6 Image Signal Processor

An image signal processor (ISP) is a specialized image processor in digital cameras. The goal of the ISP in a phone camera is to take the raw sensor data from the camera, and transform it to a user-visible image. Common stages of an ISP pipeline include color correction, lens correction, demosaicing and noise reduction (Buckler et al., 2017).

Many mobile device producers utilize third-party ISP chips for at least part of their ISP pipelines. This implies that different phones, even if they have similar cameras, might produce very different photos. There is very high heterogeneity in ISP designs and vendors (Rameshwar, 2019; Savov, 2017).

(a) HEIF: backpack

(d) HEIF: wine bottle

(g) HEIF: safety pin

(j) JPEG-q100: beer bottle

(b) JPEG: bonnet

(e) JPEG: wine bottle

(h) JPEG: purse

(k) JPEG-q85: lighter

(c) PNG: bonnet

(f) PNG: beer bottle

(i) PNG: stethoscope

(l) JPEG-q50: lighter

Figure 5. Example of photos that cause instability due to different compression formats.

Moreover, ISP pipelines in mobile devices have become more complex and might make different decisions on a photo based on the environment and the object being photographed (Statt, 2016; Bayford, 2019). The phone vendor typically does not provide access to raw sensor data before the ISP chips, and so even when phones allow to access to raw photos, those photos might have gone through some ISP pipeline stages.

Past work has shown that changes to the ISP can have a significant effect on deep learning efficacy (Buckler et al., 2017; Liu et al., 2015). Therefore, we would like to characterize how differences in phone ISPs contribute to instability. To the best of our knowledge, no phone provides direct access to its full ISP pipeline. For this reason we used a technique employed in prior work of using ImageMagick and Adobe Photoshop as simulated software ISPs (Buckler et al., 2017).

We converted the raw photos taken from the iPhone and Samsung phones, using a software pipeline. We then evaluated the classification on the resulting uncompressed PNG. The results are shown in Table 4.

Table 4. Accuracy and instability for images converted with ImageMagick or Adobe Photoshop

METRIC	RESULT
ADOBE ACCURACY	49.96%
IMAGEMAGICK ACCURACY	54.75%
INSTABILITY	14.11%

We can see that different ISPs resulted in 14% instability, implying that a large part of instability we saw in the end-to-end experiment may be attributed to ISP differences. Later in (§9.2) we will examine the potential of using raw images to overcome ISP and compression differences.

7 PROCESSOR AND OS

In this section, we investigate two other potential causes for instability: (1) OS differences between the phone might affect how images are loaded and operations are scheduled; and (2) hardware differences might affect floating point calculations and instruction scheduling at inference time.

In order to test these two potential sources of instability, instead of taking images with the different phones and observing the differences, we run an experiment with a pre-defined set of photos, where we simply try to conduct inference on different phone models. To this end we wrote a simple app that loads a subset of the Caltech101 dataset (Fei-Fei et al., 2004) and runs classification on a subset of the images using MobileNetV2 trained on ImageNet. We tested our app on 5 phones, shown in Table 5, using the Firebase Test Lab (fir), a service that allows developers to test their apps on different phones.

Table 5. Phones used in the Firebase Test experiment.

PHONE	SOC
SAMSUNG GALAXY NOTE8	EXYNOS 9 OCTA 8895
HUAWEI MATE RS	HISILICON KIRIN 970
PIXEL 2	SNAPDRAGON 835
SONY XZ3	SNAPDRAGON 845
XIAOMI MI 8 PRO	HELIO G90T (MT6785T)

During our experiment we observed very little instability. In fact the only two phones to produce any different predictions or confidences were the Huawei and Xiaomi phones. Both Xiaomi and Huawei produced the same exact confidences and prediction, and the rest of the phone models (Samsung, Pixel and Sony) produced consistent predictions as well, but different than the Xiaomi and Huawei phones. These difference resulted in a small amount of instability: 0.64%.

We suspect the reason for the difference in prediction was due to differences in the OS's handling of JPEG decoding, rather then differences in hardware. To confirm this we looked at the MD5 hash of the loaded JPEG images, and indeed Huawei and Xiaomi produced different MD5 hashes then the rest of the phone. We further confirmed this by running our experiment on PNG images; when running on PNG images we detected no instability across all phones. Therefore, we conclude that the processor and OS are not a major source of instability in our experimental setup.

8 Takeaways From the Experiments

To summarize, the end-to-end experiment demonstrated there is significant instability in model prediction generated by mobile phones even when they are taking photos of the same object in the same environmental conditions. Our experimental results show that there can be 14%-17% instability overall and even higher for individual classes.

We examined potential root causes of the instability. From our analysis it seems that most of the instability can be explained by the different ways mobile phone process images. We have seen that compression differences can cause about 5-10% instability, and ISP pipeline differences may cause about 14%. We saw very little instability caused by running different OSes or processors. In the next section, we evaluate how to mitigate instability across devices.

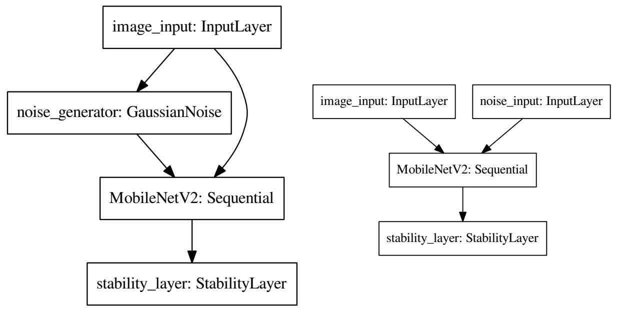
9 Mechanisms to Reduce Instability

In this section, we propose and evaluate three different approaches to reduce instability when running a model on different devices:

1. Fine-tuning the model using stability loss to be more robust to input noise.
2. Reducing noise in the input by using raw images, and applying consistent ISP and compression.
3. Modifying the prediction task to account for instability, e.g. using the top 3 predicted classes.

9.1 Stability Training

The main approach we investigated to reducing instability is fine-tuning the model to images taken by the phone. However, fine-tuning to a specific phone model introduces some challenges. A naive approach would be to train on photos taken from every phone the model might run on. However, there are thousands of constantly changing phone models, and developers often cannot anticipate which devices will run their model.



(a) Gaussian noise generator (b) Two images as inputs

Figure 6. Examples of different versions of the fine-tuned stability model. We tested different versions of noise generation and stability loss. In the embedding distance stability loss an extra dense layer is added to the model.

Therefore, an alternative solution is to use techniques that make the model more robust to noise. Robust machine learning is a well-researched area (Hein & Andriushchenko, 2017; Nguyen et al., 2015; Szegedy et al., 2014; Kurakin et al., 2017; Bastani et al., 2016; Cissé et al., 2017; Tanay & Griffin, 2016; Akhtar & Mian, 2018; Jia & Rinard, 2020; Recht et al., 2019b), but most papers deal with robustness to adversarial examples hand-crafted to change model predictions, rather then non-adversarial noise. We are inspired by prior work by Zheng et al. (Zheng et al., 2016) on stability training, which is a technique on how to improve model accuracy under noise from the environment. We further develop the stability training idea to reduce instability caused by different devices.

In stability training, during training each training image is complemented by an augmented image, which is generated by adding uncorrelated Gaussian pixel noise, i.e. if k is the pixel index for image x, and σ^2 is the standard deviation of the noise, we generate x' such that:

$$x'_k = x_k + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Because Gaussian noise is not representative of the differences between two phone images, we tried another version of noise generation: simulating the noise introduced by different phone ISPs. Our simulated phone noise randomly distorts different aspects of the training image: the hue, contrast, brightness, saturation and JPEG compression quality.

The model is trained with an augmented loss function, where θ are the trainable model weights, and α is an adjustable

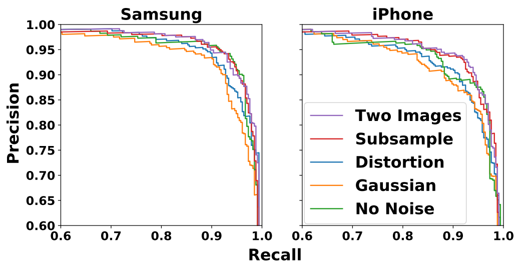
Table 6. Instability between iPhone and Samsung photos with fine-tuned MobileNetV2 with different stability losses and noise generation schemes. The hyper parameters for each training scheme are also included.

(a) Embedding distance loss.

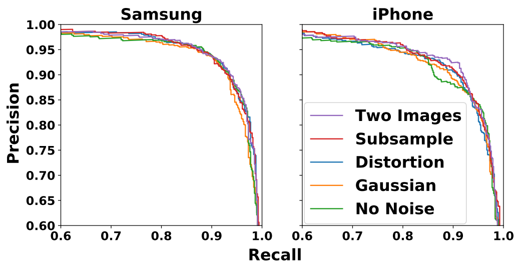
NOISE	HYPER PARAMETERS	INSTABILITY
TWO IMAGES	$\alpha = 0.001$	3.91%
SUBSAMPLE	$\alpha = 0.001$ $\#images = 10$	4.22%
DISTORTION	$\alpha = 0.01$	5.12%
GAUSSIAN	$\alpha = 0.001$ $\sigma^2 = 0.04$	5.12%
NO NOISE	N/A	7.22%

(b) Relative entropy loss.

NOISE	HYPER PARAMETERS	INSTABILITY
TWO IMAGES	$\alpha = 0.01$	6.32%
SUBSAMPLE	$\alpha = 0.01$ $\#images = 10$	5.72%
DISTORTION	$\alpha = 0.1$	4.52%
GAUSSIAN	$\alpha = 1$ $\sigma^2 = 0.025$	4.82%
NO NOISE	N/A	6.62%



(a) Embedding distance loss



(b) Relative entropy loss

Figure 7. Precision-recall curves for the different stability fine-tuning schemes, tested on Samsung and iPhone phones. Stability training not only reduces instability but slightly increases accuracy.

hyper parameter:

$$L(x, x', \theta) = L_0(x, \theta) + \alpha L_s(x, x', \theta)$$

L_0 is the regular classification cross entropy loss, i.e. for the predicted label vector y and true label vector $\hat{y}$:

$$L_0(x, \theta) = -\sum \hat{y}_j Log(P(y_j|x, \theta))$$

The stability loss $L_s(x, x', \theta)$ can come in one of two forms:

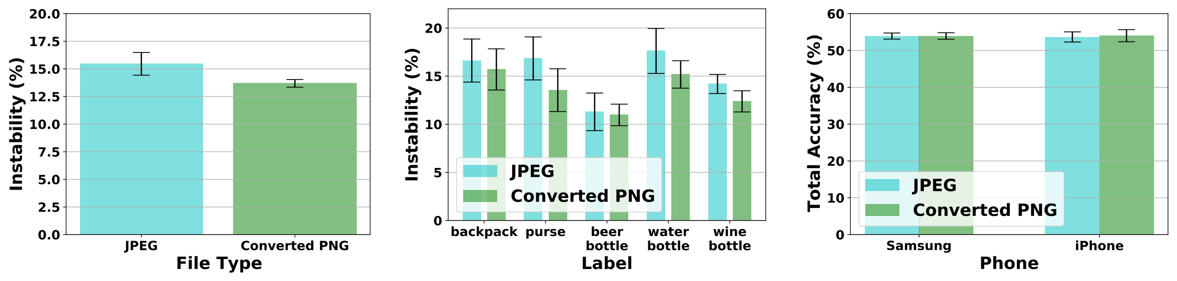
- The relative entropy (Kullback–Leibler divergence) over the model prediction of the input image against those of the noisy image:
$$L_s(x, x', \theta) = -\sum P(y_j|x, \theta) Log\Big(\frac{P(y_j|x', \theta)}{P(y_j|x, \theta)}\Big)$$
- The Euclidean distance between the embedding layer (the input to the last fully-connected layer of the model) between the input image and the noisy image:
$$L_s(x, x', \theta) = \|f(x, \theta) - f(x', \theta)\|_2$$

We implemented the stability training technique (Zheng et al., 2016) using Keras (ker) with a Tensorflow 2.3.0 backend (Abadi et al., 2016). We tried different variations of the stability training model to evaluate which approach would reduce instability the most. An illustration of some of the different versions of stability models we tried can be seen in Figure 6. We train using a Tesla K80 GPU.

We compare our noise generation models to a version of the stability training in which the noisy image is not automatically generated but rather supplied as a separate input to the model. For instance, if we are training on images from the end-to-end experiment taken by the Samsung phone, we can supply the equivalent images from the iPhone as the noisy version of the images.

In our final stability training experiment we similarly train on two images, one from Samsung and one from iPhone, but this time we limit the number of images we collect per class for iPhone. This version simulates how many images a developer would need to collect to adjust their model to a new phone. For example if we have a dataset from Samsung phones, it may be possible to augment it by collecting a limited amount of photos per class (e.g. 10) from an iPhone to make the model more stable. We treat the number of images per class as a hyper parameter. This version is similar to the two image version of our model but for every image from Samsung, instead of supplying to the model the corresponding iPhone image, we pick one image from a small subset of iPhone images for the same class.

We trained a model for each version of noise generation with each version of stability loss on Samsung photos taken from the end-to-end experiment. For the versions of the training that require noisy input images we use the photos from the iPhone. For a base model we use a MobileNetV2 pre-trained on ImageNet. We test the result on photos taken both from Samsung and iPhone. We found our hyper parameters for



(a) JPEG vs. raw converted images on Samsung and iPhone.

(b) JPEG vs. raw converted images on Samsung and iPhone, broken by class.

(c) Accuracy of JPEG vs. raw converted images on Samsung and iPhone.

Figure 8. Comparing JPEG and raw photos which where converted to PNG on iPhone and Samsung phones.

the models using grid search. We compare all the stability trained model with regular fine-tuning of the base model on the Samsung phone without a stability loss. To evaluate the embedding distance loss we added one extra fully-connected layer to the base model.

Table 6 presents the instability results across all noise generation schemes and both stability losses. We can see from the results that fine-tuning without stability training (denoted as "No noise") reduces instability the least. The results also demonstrate that including images from the iPhone for every image from the Samsung phone, and using embedding distance loss reduces instability the most. Yet, collecting many photos from each phone is not realistic.

Fortunately, augmenting the model with a modest number of photos per class from an iPhone (10 photos for each class), and using embedding distance loss resulted in fairly similar instability to collecting the entire dataset from an iPhone. We get 4.22% instability with 10 photos per class, vs. 3.91% with the entire dataset.

This solution though still requires calibration photos from each new phone we encounter. If that is not possible, using an image distortion noise with a relative entropy loss still provides a significantly lower instability (4.52%) than the baseline, and requires no new data collection.

Figure 7 contains the precision-recall graphs for all models trained under different noise generation schemes and stability losses. Interestingly, stability training has a small accuracy benefit as well as a benefit to instability. The two fine-tuning modes that augment the Samsung photos with iPhone photos provide the highest accuracy benefit.

9.2 Using Raw Images in Inference

Our second approach is to test whether using raw images might reduce the level of instability, by comparing instability between JPEG and raw images. We repeated our end-to-end experiment but this time have each phone take both a compressed JPEG image and a raw DNG image, which is then converted to PNG. This limited the experiment to only the iPhone and Samsung phones, since they are the only phones capable of taking raw images. The conversion is done in a consistent manner for both phones using ImageMagick (ima), to eliminate any differences that may arise from using different ISPs between the phones.

Figure 8(a) and Figure 8(b) compare the instability between photos of the same object on both phones using MobileNetV2. The results demonstrate that using raw images does indeed slightly reduce the instability, both in the aggregate and in every class independently, but not by a significant amount.

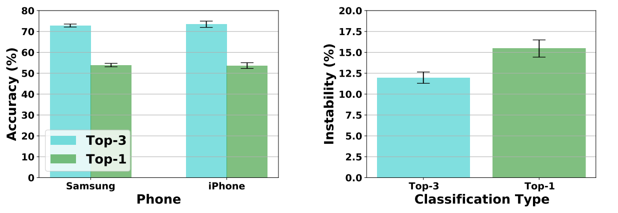
Figure 8(c) compares the accuracy of JPEG vs. raw converted images. This experiment interestingly shows that instability and accuracy are not necessarily correlated. While results on photos taken from both phones had similar accuracy, with iPhone photos producing only slightly better results, the instability was much higher than the accuracy difference. Additionally, using converted raw images did not lead to significant changes in accuracy.

Utilizing raw images and a consistent conversion pipeline results in an average 11.5% improvement in instability. Consistently throughout all experiments utilizing raw images outperformed using the phone's pipeline.

However, utilizing raw images didn't eliminate instability completely. This implies, as described in §6, that even in phones with raw image access, it is not always clear at what stage of the pipeline we get the raw image from. In addition, utilizing raw images requires hardware support from the devices. For instance in our end-to-end experiments only two of the five devices where able to take raw images.

9.3 Simplifying the Classification Task

The last approach to reducing instability is to modify the prediction itself. For many models, such as recommendation systems or document search, it might be good enough for the correct prediction to be in one of the top n predictions, rather than being the top result. To test this approach, using the same experimental setup we used before, we test the accuracy of having the correct classification appear in one of the top three results (Figure 9(a)). Unsurprisingly we achieve a higher accuracy. Similarly, in Figure 9(b) we can see that instability is also improved when we use top three results. Both accuracy and instability are improved by about 30%. Unfortunately, such task simplification is not viable for every application, and even for those where it is feasible, requiring the user to sift through additional possible classification results may degrade the user experience.



(a) Top 3 prediction accuracy for end-to-end experiment.

(b) Top 3 prediction instability for end-to-end experiment.

Figure 9. Accuracy and instability of a model that displays the top three classifications compared to only the top one.

10 Related Work

Deploying deep learning on edge devices is a widely researched topic. Due to the limited resources of the edge devices, prior work focused on different techniques to make DNN models smaller and faster. MobileNet (Howard et al., 2017; Sandler et al., 2018) and SqueezeNet (Iandola et al., 2016) are examples of models optimized to run on resource-limited devices. Another way to improve the size and performance of the models is by using techniques such as quantization (Han et al., 2016; Hubara et al., 2016; Li & Liu, 2016; Jacob et al., 2018) and pruning (LeCun et al., 1989; Hassibi & Stork, 1992; Han et al., 2015; 2016). However, these prior works do not consider the variability of running the model across different devices.

Prior work from Facebook (Wu et al., 2019) presents the challenges of deploying machine learning algorithms on heterogeneous environments, such as mobile phones. They focus on the hardware challenges of accelerating ML on the edge, while maintaining performance, accuracy and high user experience. Their work is complementary to ours, as it mostly focuses on the latency and throughput of the model.

Past work examined the effects of image signal processors (ISPs) on deep learning accuracy. Buckler et al. (Buckler et al., 2017) evaluated the effects of different stages of the ISP on accuracy and power usage, by creating a reversible ISP pipeline simulation tool. Using this analysis they created a low-power ISP for deep learning. Liu et al. (Liu et al., 2015) observe that perceived quality of an image is not correlated with accuracy of a CNN. Based on this insight they create a low-power ISP. Schwartz et al. (Schwartz et al., 2018) built an end-to-end ISP using a CNN by learning a mapping between raw sensor data and ISP processed images. To summarize, this set of works attempts to design a new ISP pipeline for deep learning. In contrast, our work treats ISPs as a black box, as they are largely outside the control of the developer and not consistent across devices.

There is a large body of work on deep learning robustness (Nguyen et al., 2015; Szegedy et al., 2014). Akthar et al.survey the different papers in this area (Akhtar & Mian, 2018). The majority of work on DL robustness focuses on noise from adversarial examples, which are images specifically designed to produce incorrect results in models (Kurakin et al., 2017; Bastani et al., 2016; Cissé et al., 2017).

In contrast to work on adversarial robustness, DeepXplore (Pei et al., 2017) and DeepTest (Tian et al., 2018) focus on how to train models to be more robust to environmental noise that may arise while driving, such as foggy road conditions. In our work we show that the device itself might add instability to the model. DeepCorrect (Borkar & Karam, 2019) focus on how to reduce Gaussian noise created during image acquisition. Meanwhile our work focuses on differences in noise between different devices. Dodge et al. (Dodge & Karam, 2016) study how models are affected by random Gaussian noise. Our fine-tuning model is inspired by prior work on stability training (Zheng et al., 2016), which tries to make models more robust to small perturbations in input data. We expand the work by adding a noise model designed to simulate differences between phones. We further show how to utilize stability training with minimal data collection.

11 Conclusions

This work examines the source of variation of model inference across different devices. We show that accuracy is a poor metric to account for this variation, and propose a new metric, instability, which measures the percentage of nearly-identical inputs producing divergent outputs. We demonstrate that for classification, different compression formats and ISPs account for a significant source of instability. We also propose a technique to fine-tune models to reduce the instability due to different devices.

We believe further exploring other sources of instability is an important topic for future work. Other potential sources, which are beyond the scope of our paper, include variations in cameras and lenses, lighting and visibility conditions.

Acknowledgments

We would like to acknowledge Pete Warden for his advice on this work, and the anonymous reviewers for their helpful suggestions.

References

Firebase test lab. https://firebase.google.com/docs/test-lab.

Imagemagick. http://www.imagemagick.org/.

Keras. https://keras.io/.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Akhtar, N. and Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., and Criminisi, A. Measuring neural net robustness with constraints. In *Advances in neural information processing systems*, pp. 2613–2621, 2016.

Bayford, S. How ai is changing photography. https://www.theverge.com/2019/1/31/18203363/ai-artificial-intelligence-photography-google-photos-apple-huawei, 2019.

Boncelet, C. Image noise models. In *The Essential Guide to Image Processing*, pp. 143–167. Elsevier, 2009.

Borkar, T. S. and Karam, L. J. Deepcorrect: Correcting dnn models against image distortions. *IEEE Trans. Image Process.*, 28(12):6022–6034, 2019. doi: 10.1109/TIP.2019.2924172. URL https://doi.org/10.1109/TIP.2019.2924172.

Buckler, M., Jayasuriya, S., and Sampson, A. Reconfiguring the imaging pipeline for computer vision. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 975–984, 2017.

Cissé, M., Bojanowski, P., Grave, E., Dauphin, Y. N., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 854–863. PMLR, 2017. URL http://proceedings.mlr.press/v70/cisse17a.html.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 248–255, 2009. URL http://image-net.org/.

Dodge, S. and Karam, L. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6. IEEE, 2016.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Generative-Model Based Vision*, pp. 178–178, 2004.

Google. Optimize images. https://developers.google.com/speed/docs/insights/OptimizeImages, 2019.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1135–1143, 2015. URL http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

Hassibi, B. and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In Hanson, S. J., Cowan, J. D., and Giles, C. L. (eds.), *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pp. 164–171. Morgan Kaufmann, 1992. URL http://papers.nips.cc/paper/647-second-order-derivatives-for-network-pruning-optimal-brain-surgeon.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2266–2276. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6821-formal-guarantees-on-the-robustness-of-a-classifier-against-adversarial-manipulation.pdf.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. Speed/accuracy trade-offs for modern convolutional object detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3296–3297, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4107–4115, 2016. URL http://papers.nips.cc/paper/6573-binarized-neural-networks.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., and Van Gool, L. Dslr-quality photos on mobile devices with deep convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3277–3285, 2017.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A. G., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2704–2713. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00286. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html.

Jia, K. and Rinard, M. Exploiting verified neural networks via floating point numerical error. *arXiv preprint arXiv:2003.03021*, 2020.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJGU3Rodl.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In Touretzky, D. S. (ed.), *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pp. 598–605. Morgan Kaufmann, 1989. URL http://papers.nips.cc/paper/250-optimal-brain-damage.

Lee, J., Chirkov, N., Ignasheva, E., Pisarchyk, Y., Shieh, M., Riccardi, F., Sarokin, R., Kulik, A., and Grundmann, M. On-device neural net inference with mobile gpus. *arXiv preprint arXiv:1907.01989*, 2019.

Li, F. and Liu, B. Ternary weight networks. *CoRR*, abs/1605.04711, 2016. URL http://arxiv.org/abs/1605.04711.

Liu, Z., Park, T., Park, H., and Kim, N. Ultra-low-power image signal processor for smart camera applications. *Electronics Letters*, 51(22):1778–1780, 2015.

Miliard, M. Google, care.ai working together to develop autonomous monitoring platform. https://www.healthcareitnews.com/news/google-careai-working-together-develop-autonomous-monitoring-platform, 2019.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 427–436, 2015.

Pei, K., Cao, Y., Yang, J., and Jana, S. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pp. 1–18, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350853. doi: 10.1145/3132747.3132785. URL https://doi.org/10.1145/3132747.3132785.

Rameshwar, K. There might be quality difference on the same iphone model sold in different countries. https://mobygeek.com/mobile/here-is-easiest-way-to-identify-your-iphones-origin-country-8232, 2019.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5389–5400. PMLR, 2019a. URL http://proceedings.mlr.press/v97/recht19a.html.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019b.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

Savov, V. Arm designed an image signal processor, and that's a big deal. https://www.theverge.com/2017/4/27/15449146/arm-mali-c71-isp-mobile-photography, 2017.

Schalkwyk, J. An all-neural on-device speech recognizer. https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html, 2019.

Schwartz, E., Giryes, R., and Bronstein, A. M. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018.

Statt, N. Apple is trying to turn the iphone into a dslr using artificial intelligence. https://www.theverge.com/2016/9/8/12839838/apple-iphone-7-plus-ai-machine-learning-bokeh-photography, 2016.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Tanay, T. and Griffin, L. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

Tian, Y., Pei, K., Jana, S., and Ray, B. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pp. 303–314, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356381. doi: 10.1145/3180155.3180220. URL https://doi-org.stanford.idm.oclc.org/10.1145/3180155.3180220.

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pp. 1521–1528, 2011. doi: 10.1109/CVPR.2011.5995347. URL https://doi.org/10.1109/CVPR.2011.5995347.

Wu, C., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K. M., Isaac, E., Jia, Y., Jia, B., Leyvand, T., Lu, H., Lu, Y., Qiao, L., Reagen, B., Spisak, J., Sun, F., Tulloch, A., Vajda, P., Wang, X., Wang, Y., Wasti, B., Wu, Y., Xian, R., Yoo, S., and Zhang, P. Machine learning at facebook: Understanding inference at the edge. In *25th IEEE International Symposium on High Performance Computer Architecture, HPCA 2019, Washington, DC, USA, February 16-20, 2019*, pp. 331–344, 2019. doi: 10.1109/HPCA.2019.00048. URL https://doi.org/10.1109/HPCA.2019.00048.

Zheng, S., Song, Y., Leung, T., and Goodfellow, I. J. Improving the robustness of deep neural networks via stability training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4480–4488. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.485. URL https://doi.org/10.1109/CVPR.2016.485.