Automated Testing of Graphics Units by Deep-Learning Detection of Visual Anomalies

Lev Faivishevsky **Ashwin K Muppalla** **Ravid Shwartz-Ziv** **Ronen Laperdon**

Benjamin Melloul **Tahi Hollander**

Amitai Armon
Intel, Visual Processing and IT Advanced Analytics Groups

Abstract

We present a deep-learning based system we developed, which performs real-time detection of diverse visual corruptions in videos. It is applied to validating the quality of graphics units in our company. The system is used for several types of content, including movies and 3D graphics. A reference video is not available during this stage of tests, due to the randomness in movie ads, game actions, website updates etc. Developing this system involved challenging data-science aspects to enable detection of small distortions with low false alert rates. We describe the full detection system, including the hardware and software aspects, and focus on the modeling approaches used.

1 Introduction

A computer graphics unit (GPU) is responsible for a feed of images to a display, such as a computer monitor. The validation of its work requires checking both hardware and software parts (e.g. drivers), with diverse systems (e.g. screens/resolutions) and diverse content, including randomly selected content. A missed defect results in visual anomalies (corruptions) in the displayed output, which map to known common types. Conventional techniques for detecting video corruptions rely on humans to look at a display and identify anomalies. This manual detection is prone to errors, since a person can be easily distracted and has a limited attention span. Automatic validation methods that are currently in use are reference-based, requiring an exact reference content that is replayed on the test system and compared with the known results. This leads to lack of scalability and non-comprehensive testing, since real-world content is not fixed and thus expected good outputs cannot be pre-profiled in a scalable way. The content in popular computer-games and websites keeps evolving, and also includes randomness, e.g. in the selection of ads or in the probabilistic outcomes of actions in games. We describe a deep learning system for real time detection of graphics unit malfunctioning. The system analyzes visual content displayed by the tested GPU, and does not assume having a reference for it. Deep learning algorithms detect visual anomalies, focusing on known possible malfunction types, and automatically alert with a predefined false alarm rate. We review the system aspects and describe in detail the developed algorithms, including a new method for the flickering [3] corruption detection.

While there was previous deep-learning work about evaluating natural image quality (see, e.g., [1], [2] and [3]), our setting is different from previous work. It requires classification of both videos and 3D-games, while the content is not known in advance, and millions of images must be processed per day (posing strict requirements on both run-time and false alert rate). Also, the issues to detect are the variety of issues seen in faulty GPUs, not the previously considered camera or communication distortions.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

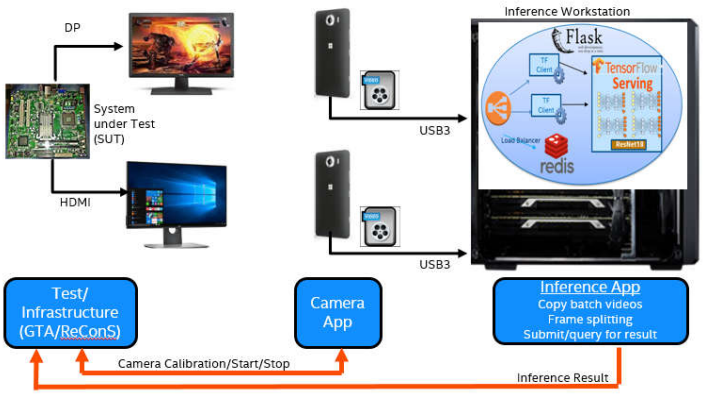

Figure 1: System operating scheme

The rest of the paper is organized as follows. Section 2 provides an overview of the system aspects. Section 3 reviews the data collection approach. Section 4 describes the developed algorithms. Section 5 provides numerical results

2 System overview

The project required building an end-to-end automated system (see Figure 1), which tests all the parts of the image processing flow. The system captures the display using a camera, since screen grabbing methods do not enable testing the content on the actual displays used. The captured videos are split into frames, and then analyzed using deep neural network models. Some models analyze single frames, while others analyze sequences of frames to identify temporal distortions (such as flickers).

The camera captures the screen in a controlled environment (black boxes), free from external light interference such as shadows and reflections. Every two cameras are connected to a single desktop, which runs the data processing (converting videos to frames) and the inference engine (deep learning classification). The inference engine software was built to efficiently manage the inference of multiple models with different batch sizes, and was described in [5]. This was essential as each system analyzes about 2M frames per day in real time.

3 Data collection

The project started by the collection of labeled data to enable higher detection with a low false alert rate (crucial as each system processes 2M frames per day). Samples of various corruption types can be seen in Figure 2. Since the appearance of real cases of corruption was relatively rare (once in a few weeks), we collected corrupted examples by reproducing driver bugs that were observed in the past. For each corruption type about 5 minutes were reproduced, including different types of content and equipment. Movies were split to frames (30 per second) and manually labeled.

For each movie with a reproduced corruption, we also captured an exact reference movie. This was done to prevent the model from learning the content of the corrupted movie as a part of the corruption manifestation, instead of the disturbance itself. We indeed observed that absence of reference movies led to alerts on valid content that appeared in the training in movies with reproduced corruptions

We also captured a large amount, about 15 hours, of diverse good uncorrupted data to ensure high variability of the content in the dataset.

Figure 2: Examples of visual corruptions

4 Modeling approaches

We experimented with a number of modeling approaches to enable both accurate detection and real-time computational efficiency. Our main focus was supervised learning. We first applied several Deep CNN topologies to single frames, either tuning pre-trained models or training from scratch. We also tried to build Shallow CNN topologies and apply them either to the full images or to image patches. We tried these approaches both in multiclass modeling and in modeling each corruption separately, with different balances between the classes (depending on the amount of good data used in addition to the reference data). We additionally used CNN with weaker supervision, considering several labels of corruption types together. To tackle corruptions that manifest themselves as anomalous sequences of frames,we employed a number of time-based methods, including Time-based (3D) CNN and LSTM [6] on a sequence of features generated from images.

We additionally applied a number of unsupervised methods to cope with corruptions that were not included in the captured dataset. This included variational autoencoders (VAE) [7] trained on the good data and anomaly detection on engineered image features (see below).

We also experimented with different data augmentations, different methods of good data sampling, and different image resizing (to optimize the model accuracy and performance). As described below, currently the system uses an ensemble of different models, including Deep CNN, Time-Based CNN, LSTM on image features sequences and anomaly detection on image features. Due to space limits, we focus our presentation on our production models and their results.

4.1 Deep CNN

As a basic detection model for the system we use ResNet 18, which is known to provide a good balance between accuracy and run-time [8] and worked well in our experiments as well. The topology is employed per each captured frame. We found that the best performance is obtained by training ResNet 18 from scratch on the collected training data (rather than by pre-trained models which tried to identify objects). We tuned the thresholds for class probabilities for each corruption type using the videos of the validation dataset, to achieve 0.1% false positive rate. The shift augmentation and L2 regularization were found to be the most useful to cope with the overfitting.

4.2 LSTM on image features sequences

One of the most important corruptions to be detected is flickering, which manifests itself as short time change of image content and/or brightness. Usually at least one frame in the sequence has a degenerated content, such as constant pixel values, very dark or very bright pixels etc. The complexity of flickering detection comes from the fact that each individual frame in the flickering sequence remains regular per itself, since, for example, we cannot a priory define low image brightness as a corruption.

We developed a new approach to analyze such short time degenerations of the video sequence content. We employ the entropy of image distribution as a measure of degeneration of its content. During the flickering sequence, the entropy of frames varies significantly, whereas the entropy of frames in a regular video sequence changes slowly. As the flickering sequences repeat themselves during a film, there is a pattern associated with this corruption. This enables to train a model to distinguish it from the regular sequences, which may include scene changes, resulting in sharp differences in entropy

of frames as well. Direct computation of the entropy of image distribution is a computationally expensive task, therefore we apply an estimate of the entropy of the frame pixels. As a JPEG compression may provide a bound on the image entropy [4], we use the size of the image file after JPEG compression as a measure of image degeneration. Indeed, we observed significant variations in the jpeg file sizes in the flickering sequences.

Across different SW and HW causes of the flickering corruption, the frequency and the exact visual form may vary. Therefore, we need to train a robust model to enable a robust corruption detection of such video sequences. We applied 2-layer LSTM for the detection of the flickering, where the inputs to the model are sequences of entropies (JPEG sizes). We found that the optimal tradeoff between the accuracy of the flickering detection and the computation load on the system is achieved for sequences of 32 frames. Therefore we use sequences of 32 image jpeg sizes as an input to LSTM, and the output is the probability of flickering occurrence in this window. The optimal threshold for alerting about the flickering presence is tuned to ensure a low false alarm rate of the system, below 0.1%

4.3 Time Based CNN

To detect complex corruptions having not only a spatial pattern but a complicated temporal structure, we use a Time Based CNN model, in which the CNN is applied to a sequence of *N* frames of RGB images (3 channels) simultaneously. We found that the best performing topology is based on ResNet 18, which is modified to accept as an input an image of *3N* channels, obtained by sticking together all frames of the sequence. We perform shift augmentations on the modified images together. During the training we use overlapping sequences to maximize the amount of the data, but in the inference time we use non-overlapping windows only to ensure a high computational efficiency of the system.

4.4 The unsupervised models

For the unsupervised algorithms, aiming to detect previously unseen corruptions, we try different approaches:

- **Variational auto-Encoder (VAE)** We used the VAE algorithm in order to detect anomalies in our data. Variational auto-encoders simultaneously train a generative model $p_\theta(x,z) = p(x|z)p_\theta(z)$ for data x using auxiliary latent variables z, and an inference model $q_\theta(z|x)$ by optimizing a variational lower bound to the likelihood $p_\theta(x) = \int p_\theta\left(x,z\right)dz$, where the generative model p_θ is modeled as Gaussian - $p_\theta\left(x|z_1\right) = \mathcal{N}\left(x|\mu_\theta\left(z_1\right),\sigma^2_\theta\left(z_1\right)\right)$. We trained the VAE based on the movies without corruptions, and used its inference model to predict 'good' frames. At inference time, we use the log-likelihood of the model to predict if the input frame is from the same distribution as the training one. We choose the threshold for this decision based on unrelated validation data.
- **Energy-Based model** In this model we split each frame into 32x32 pixel patches and calculate the Energy for each patch: $E = \sqrt{\sum r^2_{i,j,m}}$ where $r_{i,j,m}$ is the value of the patch at the position (i,j) in the m-th channel, normalized by the patch mean. Then, we calculate the average of the lowest k patches (where k is a hyper-parameter, currently set to 10). The final decision is made by comparing this average to a threshold that is selected based on unrelated validation data. In order to be agnostic to a wide range of recording conditions (such as illumination, different cameras, screens, etc.), we normalized each patch. We tried several different approaches for normalization, and the best normalization method was dividing by the average energy of the corresponding patches in three preceding frames.

5 Results

We measure the overall detection performance of the system in terms of recall for corrupted frames (or frame sequences for time based models) for a specific false alert rate of 0.1%. This value ensured a significant reduction of manual review required for the system operation (about 20X), which provided a significant benefit for our customers. The system achieved more than 80% of recall on average, in terms of detected corrupted frames (or sequences) overall. A malfunctioning of a graphics units usually manifests itself in a significant number of corrupted frames, therefore our application-wise detection capability was substantially higher.

The summary of detection results per corruption is presented in Table 1, where we also compare the performance of our supervised models with the recall of the selected unsupervised method. To illustrate the benefits of training a model from scratch that we adopted in this project we put the detection results of finetuned ResNet 50 in this table as well.

For a set of complicated corruptions of Triangles appearances, Green flash and Blankout we considered both Deep CNN (frame detection) and Time Base CNN (sequence detection), see Table 2. The time based model obtained a significantly higher recall for the explored cases.

We additionally present a comparison of the performance of our unsupervised methods in detection of corruptions, see Table 3. In addition we provide here results of a straightforward detection method, that would follow a classical unsupervised learning algorithm scheme of an appropriate feature extraction method followed by a general anomaly detection scheme. We used the C3D [9] method for feature extraction from short video sequences and Isolation Forest [10] anomaly detection. The Energy based method achieved the best performance.

Table 1: Performance of supervised and unsupervised models per corruption. Recall (%) for FPR of 0.1%

Corruption	Unsupervised	FineTuned ResNet 50	Our supervised model
Flicker	4.0	82.4	90.1
Triangle	1.4	19.1	25.3
Display stride	1.2	93.3	99.3
Vertical lines	5.2	85.7	99.4
Horizontal lines	0.9	90.9	99.4
Green flash	1.5	36.9	63.3
Color space change	14.1	79.0	79.5
Message Popup	11.0	80.5	93.1
Macro block	6.3	51.1	67.0

Table 2: Performance of Deep CNN and Time Based CNN models per corruption. Recall for FPR of 0.1%

Corruption	Deep CNN Recall (%)	Time Based CNN Recall (%)
Green flash	52.5	63.3
Triangle	5.2	25.3
Blankout	0.1	74.9

Table 3: Results of unsupervised models - Recall for FPR of 0.1%

	Average recall (%)
Energy-based w/o normalization	6.1
Energy-based w normalization	11.4
VAE	8.3
Classical	0.7

6 Summary and Future work

We developed a new deep learning based end-to-end solution for visual corruption detection. This provides significant value, detecting issues fast with low human effort. Our future work includes unifying and compressing models to ease the computational workload, and experimenting with additional unsupervised time based methods for better detection of previously unseen corruptions.

References

[1] J. Kim J., Zeng H., Ghadiyaram D., Lee S., Zhang L. and Bovik A. C. (2017) Deep Convolutional Neural Models for Picture-Quality Prediction: Challenges and Solutions to Data-Driven Image Quality Assessment. *IEEE Signal Processing Magazine* **34**(6):130-141.

[2] Vega M. T., Mocanu D. C., Famaey J., Stavrou S., and Liotta A. (2017) Deep Learning for Quality Assessment in Live Video Streaming. *IEEE Signal Processing Letters*, **24**(6): 736-740.

[3] Patel R., Pandey S. ,Chirag I Patel, C. I. and Paul R. (2017) Effective Flicker Detection Technique Using Artificial Neural Network for Video. *Kalpa Publications in Engineering. ICRISET2017. International Conference on Research and Innovations in Science, Engineering And Technology* **1**(7):442-450

[4] Chandler, D. M. and Field, D.J. (2007) Estimates of the information content and dimensionality of natural scenes from proximity distributions *Journal of the Optical Society of America* **4**(24):922-941

[5] Avidan, E. (2018) Real-Time Deep Learning on Video Streams *Strata 2018. https://www.youtube.com/watch?v=FvJNVFKZJO0*

[6] Hochreiter, S., and Schmidhuber, J. (1997) Long short-term memory. *Neural Computation* **9**(8): 1735-80

[7] Kingma D. P. and Welling M. (2013) Auto-Encoding Variational Bayes. *arXiv: 1312.6114 [stat. ML]*

[8] He, K., Zhang, X., Ren S., Sun, J. (2015) Deep Residual Learning for Image Recognition *arXiv:1512.03385 [cs.CV]*

[9] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M., (2015) Learning Spatiotemporal Features with 3D Convolutional Networks, *ICCV 2015*

[10] Liu, F. T., Ting, K. M., Zhou, Z. H. (2008). Isolation forest. *ICDM'08. Eighth IEEE International Conference on Data Mining*