Optimizing NSFW Content Detection Using Quantization Techniques for Faster Inference

Harsh Shah, Pavan Pandya, Dhruvil Dholariya, Dev Patel
Indiana University Bloomington

Abstract
This report outlines the development of a deep learning-based classifier for detecting Not Safe For Work (NSFW) content, addressing two primary objectives. The first objective is to extend the scope of NSFW classification beyond the conventional focus on nudity to encompass additional categories, including nudity, drugs, violence, and gore. The second objective is to enhance the efficiency of the model by integrating quantization techniques into the training pipeline. Traditional deep learning models, while demonstrating high accuracy, often face limitations in real-time applications due to high latency and computational demands. To mitigate these challenges, we employed quantization-aware training to reduce the model's size and computational complexity, thereby achieving significant improvements in inference speed without compromising classification accuracy. This study examines the trade-offs between model performance and computational efficiency, providing a comprehensive evaluation of quantization's impact on both classification accuracy and latency. Experimental results reveal that the quantized model achieved a size reduction of approximately **75%** compared to its unquantized counterpart while maintaining a classification accuracy of **95%**. These findings demonstrate the viability of combining deep learning and quantization for developing robust and efficient NSFW content classifiers, making them well-suited for real-time deployment in scenarios demanding rapid and reliable content moderation.

Contents

1 Introduction

The rapid proliferation of digital content sharing platforms has created an unprecedented challenge in content moderation, particularly in the detection of ***Not Safe For Work (NSFW)*** material. As user-generated content continues to flood these platforms at an ever-increasing rate, the need for efficient, accurate, and scalable NSFW detection systems has become critical. Traditional content moderation methods, which rely heavily on human intervention, have proven inadequate in coping with the sheer volume and velocity of content uploads, necessitating the development of automated solutions that can process and filter content in real-time.

While traditional machine learning models and convolutional neural networks (CNNs) have been widely used for NSFW classification, recent transformer-based models and modern CNN architectures have shown promising results in image classification tasks. These advanced models have demonstrated remarkable accuracy in identifying inappropriate content across diverse datasets. However, the computational intensity of deep learning models often poses significant challenges for real-time applications, particularly in environments with limited resources or high-throughput requirements. This trade-off between accuracy and efficiency has become a central issue in the deployment of NSFW detection systems in practical scenarios.

Recent research in the field has primarily focused on improving the accuracy of NSFW detection models through the development of more sophisticated architectures, utilization of transfer learning techniques, and creation of larger, more diverse datasets for training. While these advancements have yielded impressive results in terms of classification accuracy, they often come at the cost of increased model complexity and computational demands. This trend has highlighted the need for optimization techniques that can maintain high accuracy while reducing the computational burden of these models. An additional limitation of most existing approaches is their predominant focus on classifying NSFW content solely based on nudity, which does not encompass the full spectrum of NSFW categories. While it is widely acknowledged that nudity constitutes a significant proportion of NSFW content on the web, it is not the sole category within this domain. Other forms of NSFW content, such as depictions of drug use, violence, and gore, are often overlooked, despite their relevance and prevalence.

Our research aims to address this critical gap in the current landscape of NSFW content detection, firstly by trying to detect multiple NSFW categories such as nudity, drugs, and violent and gore content and then by implementing quantization techniques to optimize deep learning models. We explore both post-training static quantization and quantization-aware training approaches to strike a balance between accuracy and inference speed. By reducing the precision of numerical representations in neural networks, quantization has shown promise in decreasing model size and improving inference speed without substantial accuracy loss. This study contributes to the field by addressing the practical challenges of deploying deep learning models in real-world content moderation systems, where both accuracy and speed are paramount, paving the way for more widespread adoption of efficient, deep learning-based content moderation systems in resource-constrained environments.

2 Related Work

2.1 Existing Work

The paper "*State-of-the-Art in Nudity Classification: A Comparative Analysis*" provides an in-depth evaluation of current techniques used for nudity classification in images, particularly for content moderation purposes. It examines the performance of CNN-based models, vision transformers, and popular open-source safety checkers like those from Stable Diffusion and LAION. The study highlights the limitations of existing evaluation datasets, emphasizing the need for more diverse and challenging datasets to improve model effectiveness. By comparing various models across different datasets, including Adult Content, NudeNet, and LSPD, the paper underscores the importance of continually enhancing image classification systems to ensure user safety on online platforms. This analysis is crucial for developing culturally-sensitive and accurate content moderation tools.[1]

The paper "*A Comparative Study of Tools for*

Explicit Content Detection in Images" provides a detailed evaluation of various tools and methodologies for detecting explicit content in images. It compares traditional computer vision techniques, such as skin detection using color models like HSV and YCbCr, with modern deep learning approaches, including transfer learning and fine-tuning. Notable tools evaluated include the NSFW model, which categorizes images into explicitness levels with 93% accuracy, and NudeNet, which achieves 95% accuracy using ResNet-50 architecture. The study highlights the strengths and limitations of these methods, emphasizing the challenges posed by dataset differences and resolution variations. This comprehensive analysis underscores the trade-offs between classical algorithms and advanced neural networks in explicit content detection tasks. [2]

The paper "*Auditing Image-based NSFW Classifiers for Content Filtering*" investigates the performance and fairness of three widely used NSFW image classifiers: NSFW-CNN, CLIP-classifier, and CLIP-distance. It evaluates these classifiers across diverse datasets, focusing on their effectiveness in detecting inappropriate content and analyzing biases related to demographic factors such as gender, skin tone, and age. The study highlights significant disparities in misclassification rates, emphasizing the importance of addressing these biases to ensure equitable outcomes in content moderation systems. This work underscores the need for more inclusive datasets and robust evaluation methods to improve the reliability and fairness of NSFW detection models in real-world applications. [4]

The paper titled "Building a NSFW Classifier" by Lucas Ege and Isaac Westlund from Stanford University presents a sophisticated approach to classifying images as either suitable for work (SFW) or not suitable for work (NSFW). The authors developed a classifier using Convolutional Neural Networks (CNNs) and Residual Neural Networks (RNNs) under a modified ResNet50 framework. Their model achieved a high binary classification accuracy of 93.4% and a multiclass accuracy of 88.8% on their dataset, outperforming Google's SafeSearch API, which achieved 78% accuracy on the same dataset. The study expands beyond traditional pornographic content classification by also categorizing images depicting gore and weapons. This comprehensive approach addresses the need for more nuanced content moderation on social media platforms, which is crucial for maintaining advertiser-friendly environments. The authors also explored the use of DenseNet architectures and found potential for further improvements in classification performance if trained to convergence, highlighting the importance of data quality and computational resources in deep learning applications. [3]

2.2 Quantization and its types

Quantization is a technique used to optimize deep learning models by reducing the precision of numerical representations in neural networks. This approach has shown promise in decreasing model size and improving inference speed without substantial accuracy loss. Quantization can be applied to weights, biases, and activations, often reducing them to 8-bit integers, although systems with even smaller bit widths, such as binary neural networks, have also been explored.

Static quantization and dynamic quantization are two primary approaches to post-training quantization. Static quantization involves pre-computing quantization parameters (QPs) and applying them during inference. While this method is computationally efficient, it may lead to an accuracy drop. Dynamic quantization, on the other hand, calculates QPs at runtime, offering better accuracy but with increased computational overhead.

Quantization-aware training (QAT) is an advanced technique that incorporates quantization directly into the training process. This approach allows the model to learn to perform well with reduced precision from the outset, often resulting in better accuracy compared to post-training quantization methods. QAT enables the model to adapt its parameters to compensate for the reduced precision, potentially mitigating the accuracy loss associated with quantization.

Each quantization approach has its strengths and limitations. Static quantization offers the fastest inference but may sacrifice accuracy. Dynamic quantization provides a balance between accuracy and speed but requires more computational resources at runtime. Quantization-aware training often yields the best results in terms of maintaining accuracy while reducing model size and improving inference speed, but it requires retraining the entire model, which can be time-consuming and resource-intensive.

In this project, we have implemented two quantization techniques from the ones mentioned above: Quantization-Aware Training (QAT) and Static Quantization or Post-Training Quantization (PTQ).

3 Experiments, Evaluation & Results

3.1 Dataset Preparation

For our NSFW content detection model, we curated a comprehensive dataset comprising five distinct classes: safe, nudity, drugs, violence, and gore. This diverse collection aimed to cover a wide spectrum of potentially inappropriate content, as well as safe images for contrast.

We leveraged several existing datasets and employed rigorous manual verification to ensure the quality and relevance of our data. For the "nude" category, we utilized the publicly available NudeNet dataset [6], carefully selecting and verifying 5,000 images. The "gore" [8] class consisted of 4,573 images sourced from the Roboflow Universe collection. Our "violence" [5] category was derived from video data, for which we developed a custom script to extract frames at 15-second intervals, resulting in 10,924 images. The "drugs" [7] class, also sourced from Roboflow Universe, contributed 6,400 images. Lastly, the "safe" category was a combination of images from NudeNet-Safe and non-violent content from the violence dataset, totaling 13,079 images. The "safe" category was further subdivided, with the training set comprising 3,500 images from NudeNet-Safe and 5,500 from non-violent sources, while the test set included 1,500 NudeNet-Safe images and 2,579 non-violent images.

To ensure data integrity, we manually verified each class, reviewing each image for relevance and removing any that did not align with the intended category. This human-in-the-loop approach significantly enhanced the dataset's quality and reduced potential biases or mislabeled samples.

Table 1: Dataset Distribution

Category	**Train Images**	**Test Images**	**Total Images**
Nude	3,500	1,500	5,000
Gore	3,951	622	4,573
Violence	7,000	3,924	10,924
Drugs	4,480	1,920	6,400
Safe	9,000	4,079	13,079
Total	**27,931**	**12,045**	**39,976**

To standardize the dataset, we implemented several preprocessing steps. All images were converted to PNG format for consistency. We calculated the most common dimension across the dataset and resized all images to 320x320 pixels, ensuring uniform input size for our model. Additionally, we standardized the color format to RGB, which was the most common color format across the dataset.

To facilitate efficient data loading during training, we developed a custom dataset class and dataloader. So, what the default PyTorch dataloader does is, let's say, for a batch of 64, it fetches each item individually from the disk and then processes it. This time, we have the whole dataset on the disk, and for a batch of 64, we just return it by slicing it from the original dataset. This has significantly accelerated the training process. So, our dataset class implemented on-the-fly loading of preprocessed tensor data, while the custom dataloader incorporated shuffling capabilities to enhance training randomness.

The preprocessed images along with their labels were then organized into subfolders named after their respective classes within separate train and test directories. To facilitate efficient data loading and augmentation during training, we developed a custom dataset class and dataloader. Our dataset class implemented on-the-fly loading of preprocessed tensor data, while the custom dataloader incorporated shuffling capabilities to enhance training randomness.

Our rigorous approach to dataset preparation, combining diverse sources, manual verification, and standardized preprocessing, resulted in a robust and balanced dataset. This dataset forms the foundation for training and evaluating our NSFW content detection model, enabling it to recognize a wide range of inappropriate content across various categories.

3.2 Experiments

We conducted our experiments in three main phases: baseline model implementation, post-training static quantization, and quantization-aware training. All experiments were implemented using PyTorch, leveraging its robust ecosystem for deep learning and quantization.

Baseline Model:

For our experiments, we began with a baseline model using the ResNet18 architecture, a lightweight convolutional neural network (CNN) known for its efficiency and effectiveness in image classification tasks. The architecture was customized to suit our NSFW content detection task, which involves classifying images into five distinct categories. The ResNet18 backbone was initialized with pre-trained weights on ImageNet to leverage transfer learning, ensuring faster convergence and improved performance on our dataset.

To adapt ResNet18 for our classification task, we modified the fully connected (fc) layer of the network. Specifically, the original classification head was replaced with a custom head comprising a linear layer with 512 hidden units, followed by a ReLU activation function, a dropout layer with a rate of 0.3 to prevent overfitting, and a final linear layer mapping to the five output classes. Additionally, we implemented an option to freeze the backbone (pre-trained layers) during training. This allowed us to focus on fine-tuning only the classification head while retaining the feature extraction capabilities of the pre-trained backbone.

Post-Training Static Quantization:

After training the baseline ResNet18 model, we implemented post-training static quantization. This technique involves converting a trained floating-point model into an integer-based model to reduce computational complexity and memory usage. Static quantization requires calibration data to compute scale and zero-point values for each layer, which are used to map floating-point activations and weights into integer representations. This calibration step ensures that the quantized model maintains reasonable accuracy while benefiting from reduced precision.

In practice, we prepared a subset of our training data as calibration data for this purpose. The process involved inserting quantization modules into the trained model's graph using PyTorch's torch.quantization library. Layers such as convolutional and linear layers were quantized, while certain operations like batch normalization were folded into adjacent layers to further optimize performance. The static quantization process also included fusing operations (e.g., combining convolution, batch normalization, and ReLU layers) to reduce computational overhead during inference.

The quantized model was then evaluated on the test set to measure its accuracy and latency improvements compared to the original floating-point model. While static quantization significantly reduced model size and inference time, it introduced minor accuracy degradation as compared to the baseline ResNet18 model due to the lower precision representation.

Quantization Aware Training:

To mitigate the accuracy drop observed in static quantization, we implemented QAT. Unlike PTQ, QAT incorporates quantization directly into the training process. During QAT, fake quantization modules are inserted into the model graph to simulate integer arithmetic at both training and inference stages. This allows the model to learn weight distributions that are robust to reduced precision.

The QAT process began by preparing the baseline ResNet18 model for quantization using PyTorch's torch.quantization utilities. Similar to static quantization, operations such as convolution, batch normalization, and activation were fused into adjacent layers before inserting fake quantization modules. The model was then fine-tuned on our training dataset with standard backpropagation techniques while simulating integer arithmetic for weights and activations.

One of the key advantages of QAT is that it enables the optimization process to account for quantization effects during training. As a result, the model learns how to adapt its parameters to minimize accuracy loss caused by reduced precision. After fine-tuning, we converted the fake-quantized model into a fully quantized integer-based model for evaluation on the test set.

The QAT model achieved better accuracy compared to the PTQ model, and it went even higher than the baseline model.

3.3 Evaluation & Results

Both post-training static quantization and QAT have their strengths and limitations. Static quantization is simpler and faster to implement since it does not require retraining or fine-tuning of the model. However, it often results in minor accuracy degradation due to its inability to account for quantization effects during training. On the other hand, QAT typically achieves higher accuracy by incorporating quantization into the training process but requires additional computational resources for fine-tuning.

In terms of deployment scenarios, static quantization is well-suited for applications where quick optimization is needed without significant retraining efforts. In contrast, QAT is ideal for scenarios where maintaining high accuracy is critical despite requiring additional training time. By experimenting with both techniques, we aimed to evaluate their tradeoffs in terms of accuracy, latency improvements, and practical deployment feasibility for NSFW content detection tasks.

Our experimental results demonstrate the effectiveness of different quantization approaches in terms of both accuracy and computational efficiency. We evaluated the models based on two primary metrics: classification accuracy and inference latency.

Table 2: Classification (Test) Accuracies

Model Version	**Test Accuracy (%)**
Baseline ResNet18	94.10
Post-Training Quantization (PTQ)	93.83
Quantization-Aware Training (QAT)	95.10
CNN (built from scratch)	79.84

As can be seen from the results in Table 2, QAT not only preserved but slightly improved the model's accuracy compared to the baseline, while PTQ maintained comparable accuracy with minimal degradation. We also built a CNN from scratch that had 3 convolution layers, each followed by a batch normalization, ReLU, and max pooling layer, and finally a fully connected layer, but it was only able to achieve about 80% accuracy and also took more time to be trained as compared to the pre-trained ResNet 18 model. So, we chose to proceed with the ResNet18 model because of it being faster and much more accurate, which is understandable because it is trained on ImageNet data with over a million images.

Also, for the base ResNet 18 model, we achieved a precision and recall of 0.9428 and 0.9410, respectively, and an F-1 score of 0.9409. The ideal metric in our case is recall because we should take care of false negatives, allowing a few false positives if needed, and the recall score is quite good, which means our model is good at identifying NSFW content, which can also be validated by the confusion matrix in Table 3.

Table 3: Confusion Matrix of the Baseline ResNet18 Model

Category	**Safe**	**Drugs**	**Gore**	**Nudity**	**Violence**
Safe	3,930	34	24	61	30
Drugs	60	1,842	10	6	2
Gore	10	9	593	10	0
Nudity	209	3	37	1,246	5
Violence	149	3	8	3	3,131

Table 4: Inference Latencies

Model Version	**Platform/Mode**	**Latency (ms/sample)**
Baseline ResNet18	CPU	51.22
Baseline ResNet18	CUDA	2.35
PTQ	CPU	21.90
PTQ	CPU (JIT)	19.84
QAT	CPU	25.66
QAT	CPU (JIT)	22.18

The latency measurements, as can be seen in Table 4, reveal significant performance improvements through quantization. Both QAT and PTQ achieved approximately 50% reduction in inference time compared to the baseline model on CPU. The JIT-optimized PTQ model showed the best CPU performance with a latency of 19.84 ms/sample. JIT (Just-In-Time) Compilation in PyTorch refers to TorchScript, a feature that compiles PyTorch models into optimized, serializable code. By converting models into a static computation graph, TorchScript enables faster execution and easier deployment, allowing quantized models to run slightly more efficiently compared to their standard counterparts. While the CUDA-accelerated baseline model achieved the fastest overall inference time, this was an external P100 GPU which was faster compared to any of the GPUs avaiable on our daily drives especially mobile devices, where the quantized models' CPU performance makes them particularly valuable for deployment scenarios where GPU resources are unavailable.

Ultimately, using our entire quantization approach, we were able to reduce the model size by nearly 75%, from 45.85 MB of the base ResNet18 model to 11.49 MB of our QAT and PTQ models, which is where the major gains are achieved. This lighter model, combined with faster inference speeds and comparable (or, in the case of QAT, even better) accuracies, is best suited for real-world use case scenarios, particularly on devices with limited computing power in terms of GPU.

4 Conclusions

4.1 Conclusion

Our work demonstrates the successful implementation of quantization techniques for optimizing NSFW content detection models. Through careful experimentation with both post-training quantization (PTQ) and quantization-aware training (QAT), we achieved significant improvements in model efficiency while maintaining high classification accuracy. The QAT approach notably improved accuracy to 95.1% compared to the baseline's 94.1%, while PTQ maintained a comparable 93.9% accuracy. Barring comparable accuracies to the baseline model, the most important thing we were able to achieve is the 75% reduction in model size, making it much more practical for daily and real-time use case scenarios. These results validate the effectiveness of quantization techniques in preserving model performance while substantially reducing computational overhead.

Also, another significant outcome of our work is the reduction in inference latency. Both quantization approaches reduced CPU inference time by approximately 50%, from 51.22 ms/sample to 25.60 ms/sample for QAT and 21.90 ms/sample for PTQ, with the JIT-optimized PTQ achieving the best CPU performance at 19.84 ms/sample. These improvements make real-time NSFW content detection more feasible on resource-constrained devices, particularly in scenarios where GPU acceleration is unavailable.

4.2 Future Work

Future work that builds on our work can focus on incorporating more categories under NSFW and making it more robust to accommodate the increasing amount of NSFW internet content. It could also explore more complex quantization methods, such as mixed-precision quantization and dynamic bit-width allocation depending on layer sensitivity. Furthermore, exploring the application of these optimization strategies to newer architectures such as Vision Transformers may provide insights into their effectiveness across various model architectures. The development of automated quantization parameter tuning approaches could also aid in optimizing the trade-off between model accuracy and inference speed, making NSFW detection systems more deployable across a wide range of platforms and applications.

5 GitHub Repository

The link to the project repository on GitHub is: GitHub Repository

References

[1] Fatih Cagatay Akyon and Alptekin Temizel. State-of-the-art in nudity classification: A comparative analysis. *arxiv*, 2023.

[2] Adrien Dubettier, Tanguy Gernot, Emmanuel Giguet, and Christophe Rosenberger. A comparative study of tools for explicit content detection in images. *International Conference on Cyberworlds*, 2023.

[3] Lucas Ege and Issac Westlund. Building a new classifier. *arxiv*, 2018.

[4] Warren Leu, Yuta Nakashima, and Noa Garcia. Auditing image-based nsfw classifiers for content filtering. *Association For Computing Machinery*, 2024.

[5] Mohamed Elesawy Mohamad Hussein Mina Abd El Massih. Violence dataset, 2023. https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset.

[6] Bedapudi Praneeth. Nudenet, 2023. https://academictorrents.com/details/1cda9427784a6b77809f657e772814dc766b69f5.

[7] Roboflow. Drugs dataset, 2023. https://universe.roboflow.com/cbait/drugs_newest/dataset/2.

[8] Roboflow. Gore dataset, 2023. https://universe.roboflow.com/computer-vision-y8bhq/sensitive-content-2/dataset/1.