International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 05 | May - 2022 Impact Factor: 7.185 ISSN: 2582-3930

NSFW IMAGE CLASSIFICATION USING EFFICIENTNETV2

Ashish[1], Dr. Vinay Kumar Saini[2]

[1]Student, Department of Information Technology, Maharaja Agrasen Institute of Technology
[2]Associate Professor, Department of Information Technology & College, Maharaja Agrasen Institute of Technology

---

Abstract - We are in the century of the internet with the amount of data used each year reaching to 40+ zettabytes [1]. According to some research groups, over 30% of the internet users have experienced unwanted exposure to NSFW content through banner ads, pop-up ads, misdirected links or emails. Most of the content today is publicly viewable, and can be accessed from anywhere, be it the workplace, home or even school computers. But, not all of the content is appropriate for all users, especially children. An example such content is pornographic images which should be restricted to adults. Besides, these images are Not Safe For. Recently, many new convolutional neural network architectures have been successfully applied to many computer vision problems with efficient training and faster inference times in mind. One such architecture is EfficientNet.

This work proposes to classify NSFW images using EfficientNetV2.
A dataset of images is built by scraping Reddit, Imgur and Pornpics. Pretrained EfficientNetB5 model from Keras is used to first determine the best image size and then a final model is trained on 450x450 images with the final accuracy being 93%.

Key Words: NSFW, CNN, Deep Learning, EfficientNet, Image Classification.

1.INTRODUCTION

Only way to block NSFW images currently is to either manually block the image URL or rely on the moderators of that community to remove it. The first method has the downside that the user will have to block every image manually and that too after getting exposed to it, more importantly this approach can't deal with the same image but with a different URL. The second approach has the downside that some sites are not moderated at all (sites like YouTube) or lack moderator attention (sites like Facebook, Reddit) therefore it can take forever for the image to get deleted. A common downside of both of these is that the image is blocked after the exposure which pretty much defeats the point for some users, especially parents who are protecting their kids.

This leaves the only viable solution, that is using machine learning to detect NSFW images and block them on the fly. While this approach wouldn't have worked a few years ago because of lack of training data and underpowered computers, it's not the case in the current year. Even cheap mobile processors (those used in affordable laptops like Chromebook) are coming with dedicated AI accelerators and moreover, the power of traditional x86 chips have increased multiple folds. As of data, there's more training data available more than ever that too with no strings attached or compromised resolution.

2. METHODOLOGY

2.1 Data Collection

To get a model that works with a diverse number of images, a dataset with diverse images was needed. Unfortunately given the sensitivity of the NSFW data, no such dataset already exists. Therefore, we had to manually scrape data from somewhere.

Getting the NSFW data is easy as sites like Pornpics exist which are in essence a database of NSFW images.

Getting SFW data was challenging as it had to be diverse enough (we can't just take all the images of say oranges and train the model on them, it will be an Orange or NSFW classifier and not a SFW or NSFW classifier). Therefore, sites such as Reddit and Imgur were used to get the data. These sites offer an advantage that the top content is fairly decently moderated by their own AI algorithms and then human moderators. This ensures that false data is minimum.

To perform well in real life as well, a lot of data was needed so we wrote custom scripts to asynchronously scrape the image URLs and save them to a file. These URL files were then passed to *aria2c* [2] which is a fast and parallel file downloader, to get the images. At last, about 1 million URLs were scraped but only a subset of them was downloaded mainly due to time constraints and restrictions by respective websites which seem to have limitations on number of web requests per minute.

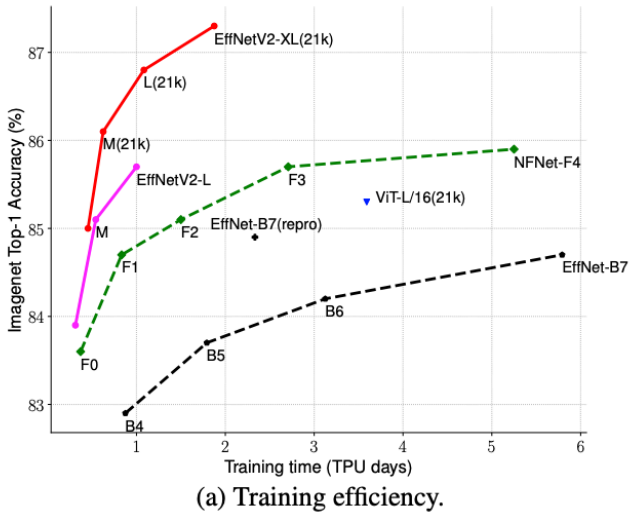
2.2 Data Cleaning

Due to the nature of sites, many images were posted multiple times and therefore downloaded multiple times. Other than the issue of duplication, there was also the issue of similar files

(such as resized images) which can't be detected by simply using traditional hashing algorithms. Luckily there's already a tool which takes care of all the data cleaning for us. It's called *czkawka* [3]. This was used to remove duplicate images, remove similar images, remove corrupted images, remove small images as well as to remove absurdly large images. After data cleaning, the final dataset had 400k images and further 10k images were selected for faster training times.

2.3 Training

Tensorflow and Keras were used as the framework of choice for deep learning. Since the task is trivial image classification, using a pre-trained model made more sense to save time and get higher accuracy. EfficientNet is the best in its class when it comes to accuracy and speed, moreover Keras already has all the EfficientNetV2 models and weights built in. EfficientNetV2L was chosen as it offers the perfect balance between the number of parameters (and thus training time) and accuracy [4] as it can be seen from figure 1.



(a) Training efficiency.

Fig -1: Training Time Vs ImageNet accuracy

As the original task of the EfficientNet model was to classify all the classes in the ImageNet dataset, the output layer had to be removed and custom layers had to be added to accurately classify the images into 2 classes (SFW or NSFW). Dense and Dropout layers were added to prevent overfitting.

Model was then compiled with the following parameters:

- loss: categorical_crossentropy
- optimizer: adam
- metrics: accuracy

As the size of images is significantly more than the RAM of the system, they had to be loaded efficiently, fortunately Keras already has a function to do this, it's called *ImageDataGenerator*. It takes care of reading the images, adding the classes, splitting the data, applying any preprocessing, shuffling and resizing. 15% of the images were used as validation and the built-in EfficientNet preprocessing function was used to preprocess the images to the requirements of the base model.

Batch size was kept fixed at 20 and the image size was varied to determine its impact on the accuracy of the final model. The following parameters were used:

- validation_split: 0.15
- dtype: float16
- batch_size: 20
- color_mode: rgb
- class_mode: categorical

To measure the impact the of the image size on the accuracy, it was varied from 100px to 500px.

Finally, the model was trained for 3 epochs on a GTX 1060 6GB laptop edition.

3. RESULT

It was observed that indeed increasing the image size resulted in higher accuracy but with diminishing returns. Figure 2 shows the relationship between the image size and the accuracy

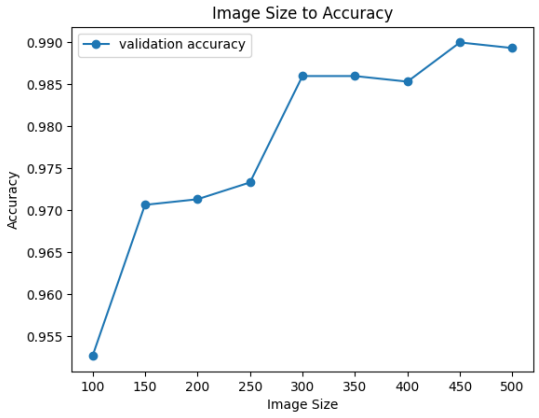


Fig -2: Image Size Vs Accuracy

The increase from 100x100 to 150x150 is over 2 percentile points but the accuracy seems to be plateauing at 450x450.

The final model was trained on full 400k images with 450px as image size. It was also noticed that 8 epochs were enough to maximise the accuracy.

Figure 3 shows the training and validation accuracy over time.

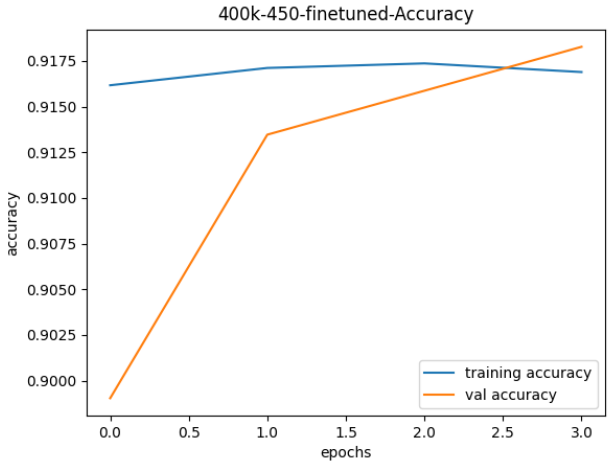
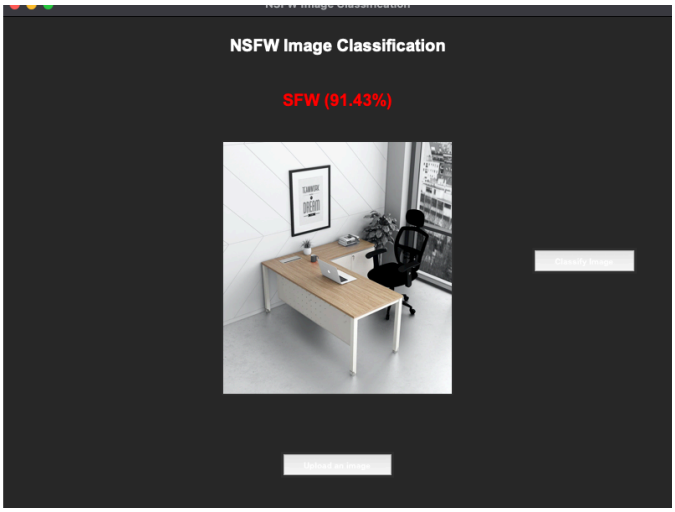


Fig -3: Accuracy over time

Finally, a GUI app using *tkinter* was made to load this model, and classify images. Figures 4 and 5 show the app correctly classifying SFW and NSFW images respectively with blurring the image if it's NSFW.

The resultant model h5 file was 480MB. This is too big to be used on lite devices.
Therefore this model was converted to a tflite model. Final size of the tflite model was just 130MB.



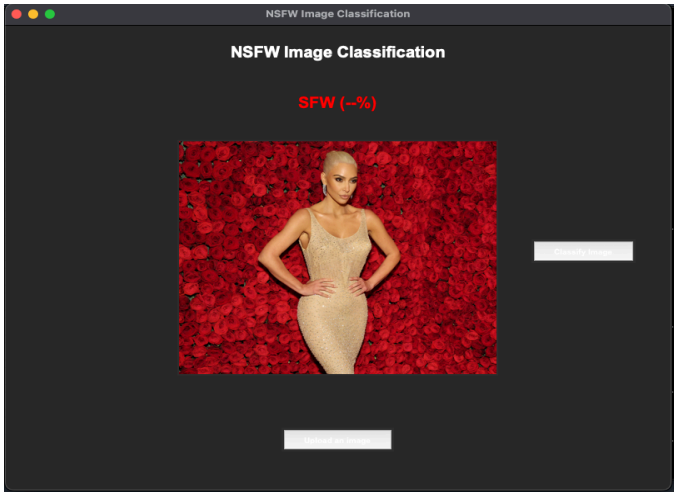
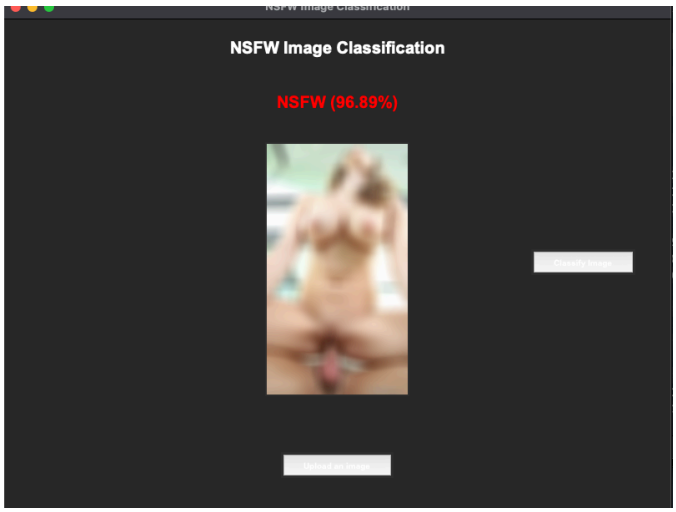


Fig -4: SFW Images



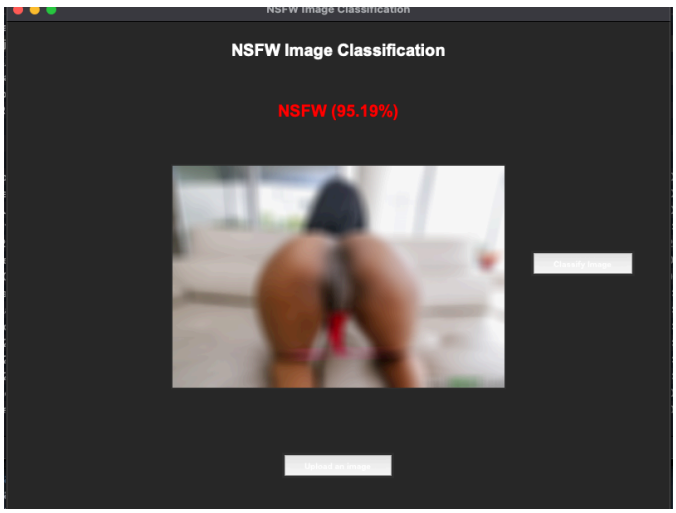


Fig -5: NSFW Images

4. CONCLUSIONS

Since the final accuracy is significantly greater than a coin toss (50%) it can be concluded that the model is capable to accurately detect NSFW images.

The model can now be paired with a browser extension like *nsfw-filter* [5] to classify images on the browser. It can be used at network level to pass all the images through this model and if they are NSFW then block them at network level for all users.

ACKNOWLEDGEMENT

I would like to take this opportunity to extend my gratitude to the following revered persons without whose immense support, completion of this project wouldn't have been possible.

I am sincerely grateful to my advisor and mentor Dr. Vinay Kumar Saini for his constant support, significant insights and for generating in us a profound interest for this subject that kept us motivated during the entire duration of this project.

I would also like to express my sincere gratitude to pukkandan, nao20010128nao, Mr.Po and nyuszika7h from *yt-dlp* discord server and Dale and RadiantXRay from *ArjanCodes* discord server for their assistance, code reviews and general best practices. Last but not the least, I would like to extend my warm regards to my family and peers who have kept supporting me and always had faith in my work.

REFERENCES

1. https://www.statista.com/statistics/871513/worldwide-data-created/
2. https://aria2.github.io/
3. https://github.com/qarmin/czkawka
4. Tan, M., V. Le, Q., & Google Research. (2021). EfficientNetV2: Smaller Models and Faster Training arxiv. https://arxiv.org/abs/2104.00298
5. https://github.com/nsfw-filter/nsfw-filter/