Udacity ML NanoDegree Capstone Project Report

Nudity / NSFW Detection In Images Using Deep Learning

Deepanshu Yadav

August 2019

CHAPTER 1

Definition

1.1 Project Overview

Internet is flooded with images. Many images posted are uploaded without any censorship. The images may belong to NSFW (Not suitable For Work). The use of censorship on images especially belonging to Nudity is still debatable as it may depend on culture to culture. But the following repercussions can still happen if such content is not actively monitored :

- The usage of nude content for prostitution and human trafficking. Many web pages are full of with ads using this content to promote such activities.
- Censoring such content can prevent underage kids to view such content. Such content is also actively promoting child pornography.

Therefore it is the job of content hosting sites to prevent such misuse by censoring.
The aim of this project to develop a deep learning model to predict whether the given image contains NSFW content or not. The project further attempts to classify the NSFW content into four categories namely :

- **Nude**
- **Semi Nude**
- **Animated**
- **Porn**

1.2 Problem Statement

As introduced earlier the goal of the project is to make deep learning model classify the image into Safe or NSFW and then if it is NSFW , it should classify into four categories namely **Nude** , **Porn** , **Semi Nude** and **Animated**. Now we will elaborate the definition for each category.

- **Nude** : Explicit exposure of genitalia of male or female , female breasts and group of people exposing their genitalia.
- **Semi Nude** : This is the most challenging category. I have defined it be men or women wearing beach wear , costumes exposing genitals like sheer clothing , obstructing genitals with something or any other image which is potentially arousing for viewer. Such kinds of images are click bates , promote prostitution and even worse child pornography.

- **Porn** : Two or more individuals engaging in sexual act though genitals may or may not be visible and every other porn category.
- **Animated** : Any image which has content similar to above categories but it is either anime or cartoon .

1.2.1 Solution

It should be clear that a rule based system wont be able to solve this problem. We got to use deep learning particularly CNN (Convolutional Neural Network). It is a two step process.

Training And Deployment

I used AWS Built in Image classification Algorithm to train the model and Aws Endpoints to deploy it in the form of API. The demo is provided in the demo folder of the repository.

Build solutions on top of the model

Build a solution on top of this API such as content filtering chrome extension or censoring tools like pury.fi. Building these solutions is out of context for this Nano Degree as this nano Degree is much more focused on Machine Learning Part. However I leave this project open source so that anyone can utilize my work to build a solution.

1.3 Metrics

I am just mentioning the metrics used to evaluate performance . The exact details , calculations and code is covered in **Metrics.ipynb.**

- **Accuracy** : As we all know accuracy is the ratio of total correctly labeled samples to the total number of samples.
- **Precision** : Fraction of relevant instances among the retrieved instances. Here we will calculate the precision of every class and we are more particularly interested when safe for work because in this case precision will be calculated of safe verses everthing that is not safe (NSFW).
- **Recall** : As stated earlier recall is more essential for safe for work class.
- **F1 Score** : As we know it is the harmonic mean of precision and recall and it takes into account both of them.

CHAPTER 2

Analysis

2.1 Data Exploration

Data Collection

The following guys had collected the data

- B Praneeth 's Data [14]
  He did an awesome job in collection of data . The data is for three classes
  - Nude
  - Sexy
  - Safe

  But the problem is I need more categories for my problem . So I made a simple tool [15] that is helpful for sub classifying the above Nude Images. You just keep all the training samples in one folder and run and run it in a jupyter notebook. I classified a few thousands of these , but then i realized that it would take a while to gather huge data. For class **Safe For Work** I sampled randomly from his huge dataset.
  Further More I also made a tool [16] that takes a screenshot of the screen and saves it into a folder. It becomes handy when you want to deliberately put difficult examples in your dataset.

  The above tools proved helpful but did not solved the problem of gathering large number of examples for training. Therefore scraping was necessary.

- Bazarov 's Dataset [17] .
  For collecting set of nude images I included the the sub category in the list he provided namely:
  - Female genitalia
  - Male genitalia
  - Breasts

  By now I had enough examples of class **nude**.

- Alex's Dataset [18].
  For classes **animated** and **porn** I scraped the data from here.

- Instagram Scrapper [19]
  For class **Semi Nude** I used his tool to scrape few Instagram pages that regularly post arousing images of men and women.

Data Organisation

The total images classwise were :

- **Nude** - 25000
- **Semi Nude** -22500
- **Animated** - 30000
- **Porn** - 25000
- **Safe For Work** - 20000

I have kept 85 percent of examples in dataset for training , 7.5 percent for validation and 7.5 for testing. They were randomly divided into three .

As discussed in the proposal report the safe for work contains mainly images of humans. So we are here developing a nudity classifier not human detector. Otherwise it will become a good human detector and not the one we wanted.

2.2 Exploratory Visualization

After collection and organization , I observed some examples in which images were useless because url from which they have to be downloaded does not exist. So I needed to remove them. Also some images were way too small and it is difficult even for human viewer to classify them. Examples are shown in figure 2.1 and 2.2.

Figure 2.1: A small image which is not clearly visible to human

Figure 2.2: The url does not exist for this image.

2.3 Algorithms and Techniques

I have used AWS built in Image Classification algorithm [11] which uses Residual Networks (Resnet)[12]. Since training a full residual takes weeks to train , i have pretrained model for quickly training it. The exact details are given in the training.ipynb.

2.3.1 What are Resnets

The core idea of ResNet is introducing a so-called identity shortcut connection that skips one or more layers, as shown in the figure 2.3. The advantage of adding this layer in very subsequent layer is that we may able to train more deeper networks. Refer this article to know more [20]. The Resnet training facility is readily availabe in aws image classification algorithm. You can even give it how much deep the network would be (for example there a lot of variants of Resnets available like Resnet -50 , Resnet -101 , Resnet -152). I chose the default value which was 152.

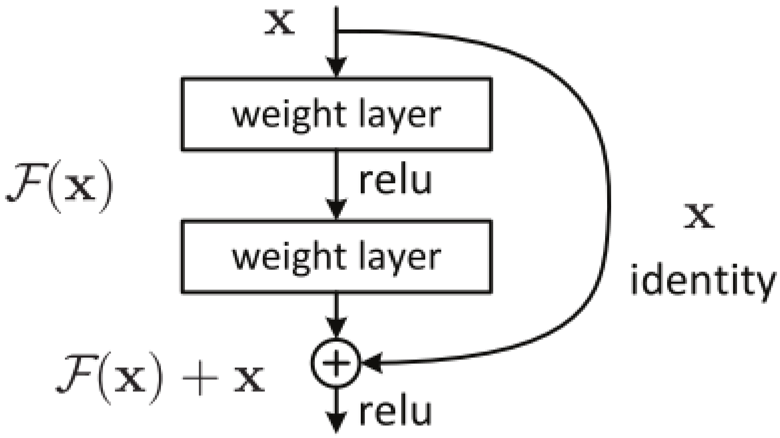

Figure 2.3: A Residual Block

2.3.2 Why I chose a ResNet

- Resnets have been known to give good results particularly in Image Net.The authors won the Image Net challenge in 2015.

- Many CNN's have known to perform classification on some irrelevant object in the background. But models like Resnets are proven to perform classification only on the relevant object. This is important in my case because in the examples the subject in present in more than 50 % of the image area.
- Many examples have occlusions but are still NSFW. Resnet are fairly invariant to occlusions.
- Resnet have little parameters than other popular models like VGG , Inception etc. So performing classification on trained model would be faster, which matches my requirements as the API should be responsive and fast.
- Aws Image classification Algorithm is based on Resnets makes much easy for me train.

2.4 Benchmark

The author at [13] has done a good job in comparing various nsfw detector api's available. I have used his dataset in order to compare my model with existing state of the art deployed models.

CHAPTER 3

Methodology

3.1 Data Preprocessing

The following steps were involved in data preprocessing .

3.1.1 Discarding anomalous images

As discussed the images whose url did not exist , which were way too small in size and which were corrupted were removed from the dataset. See Data-process.ipynb for more details.

3.1.2 Dividing into training , testing and validation

I have kept 85 percent of examples in dataset for training , 7.5 percent for validation and 7.5 for testing. They were randomly divided into three .

3.1.3 Conversion into Record IO format

The amazon built in classification algorithm requires the images to be converted into record IO format. So i have im2rec tool that comes with mxnet , then I have uploaded the converted images into S3.

3.1.4 Data augmentation

The aws classifier comes with built in data augmentation in the form of hyperparameter to the algorithm. Its name is `Augmentation_type` and i have set is value to be `crop_color_transform` which means it performs a color transform and then crops randomly from the sample.

3.2 Implementation

See training.ipynb notebook for details. The table 3.1 gives the details of base hypermeters and table 3.2 gives the best hyperparameters obtained from Aws Sagemaker hyperparameter tuning jobs. I decided to shut some tuning jobs as their performance were very similar to earlier tuning jobs.

Table 3.1: Base Hyper Parameters

Base HyperParameters	Values
_tuning_objective_metric	validation:accuracy
augmentation_type	crop_color_transform
epochs	1
image_shape	(3,224,224)
use_pretrained_model	1

I have fixed these the hyperparameters for example epochs was fixed by seeing the time taken for job to complete. It was decided epochs = 1 was enough for training jobs comparison.

Table 3.2: Tuned Hyper Parameters

Sno.	lr	batchsize	optimizer	val accuracy
1	0.00106259	24	adam	0.658315
2	0.00025830	16	nag	0.8354889
3	0.00131707	25	nag	0.8261880

Notice that job number 2 has the highest validation accuracy.

3.2.1 Problems And Their Solutions

- Training costs
  If you are training this on AWS Sage Maker , in spite using pre trained model , It takes a lot of time to train , make sure to set number of epochs low . Two was enough in my case for a p2.xlarge instance.

- Images truncated
  While training , i noticed training jobs shut down down unexpectedly draining the credits away. Then i removed truncted images from my datset and it worked fine.

- Monitoring training
  I ensured that my validation loss and validation accuracy were always comparable with training loss and validation loss. You can stop training and model artifacts(the parameters which are evaluated as a result of training) will be safe .

3.3 Refinement

After selection of best hyper parameters , i decided to train the model for a few more epochs to improve accuracy . But the training and validation plateaued at 89% and 85% respectively. So decided not to continue the training further and evaluate my model on test set using aws batch transform.

CHAPTER 4

Existing Benchmarks and Evaluation

4.1 Model Evaluation and Validation

4.1.1 Evaluation on testing Set

I used aws batch transform to apply images to the model in bulk. The result of the above batch transform was combined into a csv file(results.csv). See Metrics.ipynb and batch-transform.ipynb for details. Now as discussed the various metrics for every class obtained are shown in the table 4.1.

Figure 4.1: Confusion Matrix on the test set. _ori means original and _pred means predicted .

Table 4.1: Performance on testing set

Class Name	Accuracy	Precision	Recall	f1 score
Animated	0.985486	0.958648	0.958648	0.479324
Nude	0.852288	0.338588	0.338588	0.169294
Porn	0.89474	0.842054	0.842054	0.421027
Safe for Work	0.951144	0.738095	0.738095	0.369047
Semi Nude	0.992513	0.982892	0.982892	0.491446
Nsfw	0.972831	0.997585	0.973059	0.492584

Table 4.2: Performance with Bench Mark

Class Name	Accuracy	Precision	Recall	f1 score
Animated	0.929411	0.8	0.8	0.4
Nude	0.752380	0.071428	0.071428	0.035714
Porn	0.887640	0.583	0.583	0.2916
Safe for Work	0.663865	0.0	0.0	nan
Semi -nude	0.929411	0.833	0.833	0.4166
Nsfw	0.797979	1.0	0.797979	0.443820

4.1.2 Evaluation with benchmark

I used the dataset provided by [13] to compare my performance with existing state of the art nsfw-detection-apis. See benchmark.ipynb and analyze-bench.ipynb for details. The various metrics for every class obtained are shown in the table 4.2.

4.2 Justification

4.2.1 Points on test set

The Good Points

- From the results it is clear that the model does a very good job in classifying class **Semi-Nude** and **Animated**.

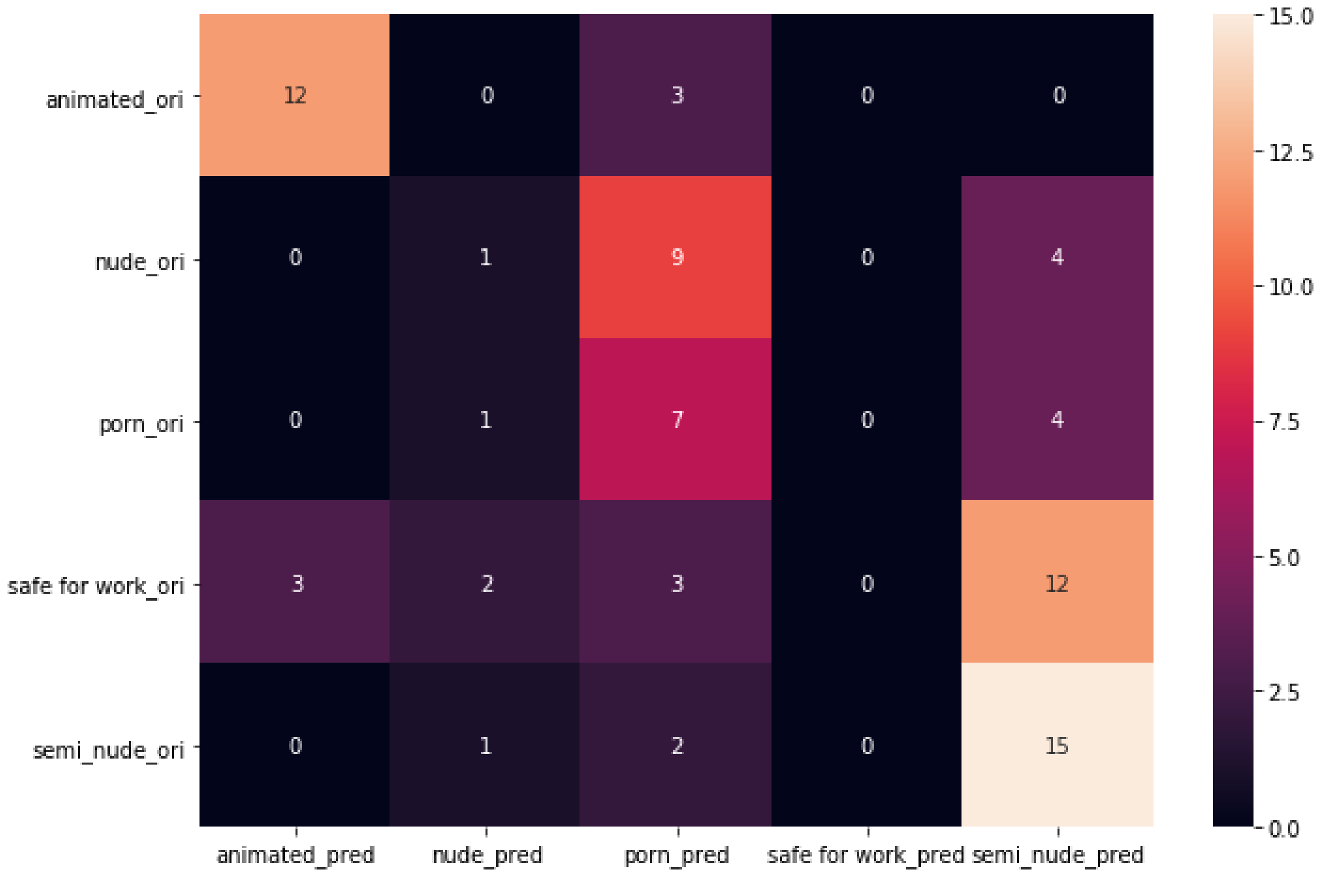

Figure 4.2: Confusion Matrix on the bench mark _ori means original and _pred means predicted .

- When all the nsfw classes were combined and evaluated against the class **Safe For Work** model metrics were better and particularly precision and recall.

The bad points

- The performance for classes **Nude** , **Porn** and **Safe for Work** could have been better.
- The model classifies class **Nude** to **Porn** and **Semi-Nude** which is understandable because there sometimes only a subtle difference between **Nude** and **Semi-Nude**. But it needs to improve on classifying **Nude** to **Porn**.
- The model needs to improve on classifying **Porn** and **Safe for work**.

4.2.2 Points on comparison with bench mark

The Good Points

- From the results it is clear that the model does a very good job in classifying class **Semi-Nude** and **Animated**.

- When all the nsfw classes were combined and evaluated against the class **Safe For Work** model metrics were better and particularly precision. Recall could have been better.

The bad points

- The performance for classes **Safe for Work** was poor. But if look closely (see the notebook analyze-bench.ipynb) many images contained examples that have defined to belong to **Semi - Nude** so the performance is bit satisfactory looking at the images in the benchmark dataset.
- Overall it needs to improve performance on **Nude** , **Porn** and **Safe For Work** .

Bibliography

[1] https://archive.org/details/NudeNet_classifier_dataset_v1

[2] http://www.cti-community.net/

[3] http://pury.fi/

[4] https://github.com/bedapudi6788/NudeNet

[5] https://github.com/bedapudi6788/NudeNet-models/

[6] https://dzone.com/articles/nudity-detection-and-abusive-content-classifiers-r

[7] https://github.com/deeppomf/DeepCreamPy

[8] https://medium.com/@praneethbedapudi/nudenet-an-ensemble-of-neural-nets-for-nudity-detection-and-censoring-c8fcefa6cc92

[9] https://medium.com/@praneethbedapudi/nudenet-an-ensemble-of-neural-nets-for-nudity-detection-and-censoring-d9f3da721e3

[10] https://github.com/bedapudi6788/NudeNet-models/tree/master/v1

[11] https://docs.aws.amazon.com/sagemaker/latest/dg/image-classification.html

[12] Deep residual learning for image recognition Kaiming He, et al., 2016 IEEE Conference on Computer Vision and Pattern Recognition

[13] https://towardsdatascience.com/comparison-of-the-best-nsfw-image-moderation-apis-2018-84be8da65303

[14] https://archive.org/details/NudeNet_classifier_dataset_v1

[15] https://github.com/deepanshu-yadav/NSFW-Classifier/blob/master/classification_tool.ipynb

[16] https://github.com/deepanshu-yadav/NSFW-Classifier/blob/master/useful_scripts/useful_scripts/

[17] https://github.com/EBazarov/nsfw_data_source_urls

[18] https://github.com/alex000kim/nsfw_data_scraper/tree/master/raw_data

[19] https://github.com/rarcega/instagram-scraper

[20] https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035