Building a NSFW Classifier

Lucas Ege
Department of Computer Science
Stanford University
lucasege@stanford.edu

Isaac Westlund
Department of Computer Science
Stanford University
iwestlun@stanford.edu

Abstract

Classification of images and video as SFW (suitable for work) vs. NSFW (not) has become vitally important today because of the rise of social media and online advertising. Advertisers have recently made it clear that they will not stand for their ads being placed on questionable or offensive content, leaving the responsibility with social media companies to maintain a "clean" environment that advertisers can feel comfortable with. Our project worked to create such a classifier using techniques and models detailed in this class to derive an effective classifier and compare to industry leaders. Utilizing Convolutional Neural Networks and Residual Neural Networks under a modified ResNet50 framework, we were able to reach 93.4% binary classification accuracy on our data set, and 88.8% multi class (4) classification accuracy. This compares to Google's SafeSearch API, which was able to achieve 78% binary classification accuracy on our data set.

Code: https://github.com/lucasege/cs230

1 Introduction

The goal of this project is to develop a classifier to interpret an input image as "suitable for work" or "not suitable for work" as well as to classify the unsuitable images based off of the inappropriate content. The categories we decided to classify are gore, pornographic content, and weapons. This is an interesting and relevant application as recently, large social media companies have been working on the same idea as they struggle to curb inappropriate images and videos posted on their platform as they fight to attract advertising. For example, Youtube recently faced criticism after one of its most well known producers, Logan Paul, posted a video of himself finding a recent suicide victim in Japan's Aokigahara Forest and Youtube's filters were unable to detect or prevent this video from being posted. Ideally, an application such as ours could be used to preemptively spot such posts before they go public.

Our model takes as input an image and outputs either a binary classification of NSFW v. SFW, or a multi class classification of the image as containing pornographic content, gore, weapons, or none of the above (SFW). The images we collected were all resized to be of size 64x64 with 3 color channels in order to allow for efficient passing through our network. We initially used a flattened vector of size (64 * 64 * 3) which we passed to our initial logistic regression for baseline results. For our future models our input features were the images themselves, which we fed into convolutional layers. We then fed this into a CNN and an RNN to output our classifications.

2 Related work

Previous open-source work largely focuses on the issue of classifying images as NSFW v SFW based entirely on pornographic content. These implementations have largely used convolutional neural networks, as they have been proven to be efficient and effective with an image classification problem. General categories for work related to this issue are general NSFW classification using CNNs, NSFW classification using RNNs, and modifications to CNNs to improve general image classification.
NSFW classification with CNNs: Mohammad Moustafa's paper, "Applying deep learning to classify pornographic images and videos," **[2]** achieves 94% accuracy on a well known benchmark dataset (NPDI pornography dataset) using a modified combination of AlexNet and GoogLeNet. The combination of two of the leading image classification models in order to correct for both of their mistakes is clever, but the added computational and storage costs are not sufficiently addressed.
NSFW classification with RNNS: In 2015 Yahoo **[1]** released an open source implementation of a NSFW classifier, this paper highlighted multiple different models applied to this task, with the highlights being their use of AlexNet and GoogLeNet (similar to Moustafa's paper), and their implementation of the ResNet50 model. The wide array of models used allowed for a quick run-down of the state of the art in image classification for this task. The best performing model on their dataset was the ResNet50 model, although their modified version of this model (Resnet50-Thin) provided a reasonable compromise between accuracy and speed.

While we appreciated the work of classifying NSFW on one category (pornography), we were more interested in an expanded multiclass image classifier, detecting other types of NSFW content, for a more well-rounded model. Thus, we looked to general image classification improvements.

General Image Classifications: Improvements to the basic usage of CNNs and RNNs have been abound in recent work, such as Iandola et. al.'s, "SqueezeNet" **[3]** which achieved accuracy akin to AlexNet with 50x fewer parameters and less than 0.5 MB of model storage. This compression of the network showed how many different CNN architectures can achieve the same level of accuracy for a task, meaning there is plenty of room to optimize for performance and storage when building a top-tier classifier. Similarly, Huang et. al.'s implementation of densely connected convolutional networks **[4]** provided a verbose model where each layer connects with all of the next layers. This implementation (DenseNet) works to eliminate the vanishing gradient problem, strengthen feature propogation all while reducing the number of model parameters. We decided to implement DenseNet for our purposes because of these reasons and our initial success for the ResNet50 framework. Simonyan and Zisserman's work with "very deep" CNNs for larger image recognition **[5]** and showed that deeper CNNs benefited the classification accuracy, allowing for "state-of-the-art" performance using just a conventional ConvNet architecture. This look into the importance of the depth of a model related to the feature input, however, was not as applicable to our data set since we used images of small size (64x64).

3 Dataset and Features

As discussed, previous implementations of NSFW classifiers have focused mostly on classifying pornography, leaving many preprocessed data sets of this class of data. However, we wanted to classify more than just pornographic images. Thus, we took the preprocessed images of pornography, provided through Kaggle **[9]**, and then looked to add images for gore and weapon classification, as well as SFW classification images. We were able to find another image set from Kaggle that provided a good amount of weapon and SFW images **[10]**. These images were of lower quality, though, so we looked to find more specific and higher-quality images. We found that Imgur provides an open API with relaxed quotas and very specific filtering. We were thus able to pull images directly from subReddits, allowing for a high degree of specificity (subReddits are often very specific) while also grabbing high quality images. Using this method, we grabbed images from /r/guns, /r/gore, /r/MedicalGore, and /r/knives.
In order to utilize both methods of data collection, we had to regularize the images to a similar format (64x64x3) to pass into our model. We then also did simple data augmentation described in **[2]** by flipping images horizontally. Using these methods, we were able to gather more than 20,000 unique images - split into 6,000 images of porn, 3,000 images of gore, 4,000 images of weapons, and around 10,000 SFW images. We then split this set of images into 90% training and 10% testing. Due to the nature of our images, showing examples would be superfluous.

These images were initially flattened for our logistic regression implementation, but were later fed directly into the network using convolutional layers.

4 Methods

Baseline

As a baseline, we flattened each image in our dataset and trained a single node with a sigmoid activation, classifying each picture as NSFW or SFW. For this model we used a basic squared loss function, $loss = \sum_{i=0}^{n} \frac{1}{n}(\hat{y}_i - y_i)^2$. As our goal was to classify an image, we decided that using various convolutional neural networks made the most sense. A convolutional neural network contains convolutional layers which are composed of n-dimensional blocks of parameters. Within these blocks, a n-1 dimensional slice of parameters, aka a filter, is used to take the convolution of the inputs. The these filters are stacked together to act like multiple nodes in a hidden layer of a standard neural network.

Convolutional Neural Networks

We tested a variety of CNN implantations, starting very simply and progressively getting more advanced and for the most part, more accurate. Our first CNN implementation consisted of five layers, a convolutional layer of shape 4x4x3x8, with a relu activation, a maxpooling layer of window size 8x8 with stride 8, a convolutional layer of shape 2x2x8x16 with a relu activation, another maxpooling layer of window size 4x4 and stride 4, and finally a fully connected layer with either 1 or 4 outputs. When using one output, the fully connected layer used a sigmoid activation to output a classification as either NSFW or SFW. For this we used a sigmoid cross entropy loss function $loss = \sum_{i=0}^{n} \hat{y}log(y) + (1 - \hat{y})log(1 - y)$. When using four outputs, we used a softmax activation to classify an image as pornographic, depicting a gore or a dead body, depicting a weapon, or SFW. For this we used a the same cross entropy loss function described above. Due to the small size of this five layer network, running and testing this network was relatively quick.

Residual Neural Networks

We next wanted to create a network much larger to attain greater accuracy. A larger model allows the network to more effectively learn complex real world data. However, if you make the model too large, there is the possibility that the network would overfit to the training set and result in higher variance. Another problem that emerges with a increasingly large models is the vanishing gradient problem. As more layers are added, the gradient becomes exponentially smaller which results in the network's learning coming to a standstill. Intuitively, the flow of data through the network is slowed with every layer added and eventually as layers are added the network becomes saturated/clogged. Currently, the standard solution to this problem is the use of a residual network. A residual network adds "skip connections" between layers. A skip connection connects adds output from an earlier layer to the input of a later layer, usually skipping two to three layers at a time, hence the "skip". Residual networks allow for data to more easily propagate from earlier layers to later layers and effectively solve the vanishing gradient problem.

We based our second model implementation around the popular ResNet50. In this implementation we used a combination of identity blocks and convolution blocks. An identity block in our model is composed of: a convolutional layer with window size 1x1, stride 1, and a relu activation, a convolutional layer with window size FxF, stride 1, and a relu activation, another convolutional layer with window size 1x1, stride 1 and relu activation function. The FxF window size of the middle layer is defined uniquely for each block. Before the output of the last convolutional layer is fed into its respective relu activation, the input of the first convolutional layer is added element wise to it. Also, to speed up the learning rate, before each convolutional layers output is fed into its respective relu activation, the output is batch normalized. Moving on, a convolutional block in our model consists of an identical set of three layers to the identity block, with the only difference being that before the input of the first layer is added element wise to the output of the last convolution, it is convolved with the same convolution as the third layer and is batch normalized. With these blocks in mind, the total network consists of: a convolutional layer, a convolutional block, 2 identity blocks, a convolutional block, 3 identity blocks, a convolutional block, 5 identity block, a convolutional block, 2 identity blocks, and finally a fully connected layer. We tested with the fully connected layer either having one output to classify as NSFW/SFW or four outputs to classify as porn/gore/weapons/SFW.

Dense Networks

For our final implementation we wanted to test a network design that we did not learn in class. This implementation was based off of a combination of our earlier ResNet50 model as well as Huang et. al.'s implementation of a densely connected convolutional network. A densely connected network is unique from a residual network, is that while a residual network will add element wise the output of an earlier layer to some later layer, a densely connected network concatenates the output of each layer to every proceeding layer. This effectively solves the vanishing gradient problem and allows for much faster learning and less memory usage compared to a standard residual network.

Our dense net uses the same convolutional block and as described in our RNN but instead of an identity block, uses a dense block. A dense block is composed of: a convolutional layer with window size 1x1, stride 1, and a relu activation function, 3 convolutional layers with window size FxF, stride 1, and a relu activation function, another convolutional layer with window size 1x1, stride 1 and relu activation. The output of each layer is batch normalized and concatenated with the outputs of all previous layers in the dense block before being fed into next layer.

We tested two sizes of dense networks. The larger network consists of: a convolutional layer, a convolutional block, 2 dense blocks, a convolutional block, 3 dense blocks, a convolutional block, 5 dense block, a convolutional block, 2 dense blocks, and finally a fully connected layer. This combines for a grand total of 74 layers. The smaller network consists of: a convolutional layer, a convolutional block, a dense blocks, a convolutional block, a dense blocks, a convolutional block, a dense block, a convolutional block, a dense blocks, and finally a fully connected layer. This combines for a grand total of 34 layers. For both of our dense networks, we only ran tests with four outputs to classify images as porn/gore/weapons/SFW.

5 Experiments/Results/Discussion

For our results metrics, we focused on the accuracy of our models in predicting a binary classification, or one of four multi class classifications.

Baseline

For our baseline we flattened all of our images and fed them to a single node with a sigmoid activation, this acheived 58.4% training accuracy, 57.8% testing accuracy. Obviously, the images are far too complex for a logistic regression.

Simple CNN

Our basic five layer CNN achieved 67.2%training accuracy, 64.3% testing accuracy when attemping to classify images as NSFW/SFW and 48%training accuracy, 44.4% testing accuracy. This model ran very quickly and as a result we could run tests until convergence quite easily. While the basic CNN perfored better than baseline and there little evidence of overfitting, the model is still not performing very well compared to our oracle and it seems the images are still too complex for the model.

ResNet50

The 50 layer RNN that we trained achieved 99.89% training accuracy and 93.4% testing accuracy classifying images as NSFW/SFW and 98.3% training accuracy and 88.8% testing accuracy classifying images as NSFW/SFW. Being a much larger network, running experiments was extremely time consuming and as a result hyperparameter testing was limited, both models were tested after 50 epochs. However, we were pleased to see that this model managed to do considerably better than Google's implementation on our data set. We concede that our network has an advantage as it was trained on our dataset while Google's was pretrained on a different dataset. While this network performed relatively well, there is evidence of overfitting considering it managed 6 percentage points higher on the training set for binary classification and 10 percentage points higher for the multiclass classification.

DenseNet

The 74 layer RNN that we trained acheived 79.65% training accuracy and 80.16% testing accuracy classifying images as porn/gore/weapons/SFW. As this network was extremely large, training and testing was extremely time consuming. While this network did not achieve better results in our tests than our ResNet50 implementation, this model was only tested after 5 epochs and was continueing to improve at a faster rate than our ResNet 50 model. There is also no apparent overfitting.

The 34 layer RNN that we trained acheived 93.6% training accuracy and 88.6% testing accuracy classifying images as porn/gore/weapons/SFW. Similar to the large version, we were not able to run this network to convergence and it was only tested for 10 epochs. However, interestingly this network actually learned at a faster rate than the large network and performed at about the same level of the ResNet50 even when the ResNet50 was trained for 50 epochs. It is very likely that if we were able to train either of the dense networks until convergence, they would be our most effective classifiers as they have much less overfitting.

Table 1: Binary Classification Accuracies

Model	**Train**	**Test**
Logistic Regression	58.4%	57.8%
Initial CNN	42.4%	40.9%
Improved CNN	67.2%	64.3%
ResNet50 Basic	56.5%	58%
ResNet50 Improved - 2 Epochs	74.4%	76.3%
ResNet50 Improved - 50 Epochs	99.89%	93.4%
Google SafeSearch (Reference)	X	78.4%

Table 2: MultiClass (4) Classification Accuracies

Model	**Train**	**Test**
Standard CNN	48%	44.4%
ResNet50 Improved - 50 Epochs	98.3%	88.8%
DenseNet74 - 5 Epochs	79.65%	80.16%
DenseNet34 - 10 Epochs	93.6%	88.6%

6 Conclusion/Future Work

For both of our tasks, we found the ResNet50 architecture provided the best results, with minor modifications. This can be attributed to multiple reasons, such as this framework's ability to prevent gradient vanishing. However, it is likely if we were able to train our dense networks to convergence, they would surpass the ResNet50 implementations. Our more sophisticated models using Convolutional blocks obviously worked better because of the ability to model the image's features more efficiently. Crucially, we found that our models did not experience large amounts of over fitting or large variance issues, as most of our train/test accuracies were similarly based.
As with most deep learning applications, the quality and quantity of data is vital, so a full scale application would require more data acquisition. We would also have liked to work with larger images than 64x64, since this would mean strictly more features and more expressiveness throughout the model, but were constrained by computational resources and wall time. We would also like to apply GANs to this field to develop a quality classifier and generator of images together.

7 Contributions

Lucas: Did most of the data collection and preprocessing to format data for the model. Integrated and tested against industry applications (Google). Built out initial Logistic regression and hyper parameter modifications to achieve results.
Isaac: Implemented the CNN and RNN frameworks, as well as later applying the DenseNet model. Also worked on hyper parameters modifications for results.

References

[1]: Mahadeokar, J. and Pesavento, G. Yahoo Engineering. (2018). Open Sourcing a Deep Learning Solution for Detecting NSFW Images. [online] Available at: https://yahooeng.tumblr.com/post/151148689421/open-sourcing-a-deep-learning-solution-for [Accessed 23 Mar. 2018].

[2]: Moustafo, M. Arxiv.org. (2018). Applying deep learning to classify pornographic images and videos. [online] Available at: https://arxiv.org/pdf/1511.08899.pdf [Accessed 23 Mar. 2018].

[3]: Iandola, F., Han, S., Moskewicz, M., Ashraf, K., Dally, W. and Keutzer, K. (2018). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1602.07360.pdf [Accessed 23 Mar. 2018].

[4]: Huang, G., Liu, Z., van der Maaten, L. and Weinberger, K. (2018). Densely Connected Convolutional Networks. [online] Openaccess.thecvf.com. Available at: http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf[Accessed 23 Mar. 2018].

[5]: Simonyan, K. and Zisserman, A. (2018). Very Deep Convolutional Networks For Large-Scale Image Recognition. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1409.1556.pdf [Accessed 23 Mar. 2018].

[6]: Tensorflow, Google

[7]: Keras

[8]: OpenCV

[9]: https://www.kaggle.com/ljlr34449/porn-data/data

[10]: Maksimova, A., Matiolański, A. and Wassermann, J. (2018). Fuzzy Classification Method for Knife Detection Problem. [online] SpringerLink. Available at: https://link.springer.com/chapter/10.1007%2F978-3-319-07569-3_13 [Accessed 23 Mar. 2018].