THE UNIVERSITY OF

MELBOURNE

Deep Learning for the Classification of Adult Images and its Utilization in Social Applications

COMP90055 & COMP90019

Computing & Distributed Computing Project

Master of Information Technology
School of Computing and Information Systems
The University of Melbourne
Semester 2, 2019

Yash Mahendra Shinde (920691)
Nitish Mathur (954892)
Nakia Silva (796504)

Supervisor: Prof. Richard Sinnott
Project Type: Software Development Project
Credit Points: 25

Abstract

The increased access to the internet coupled with the ubiquitousness of smartphones has led to a rabid usage of social applications such as social media platforms and chat applications. These applications bestow its users with the ability to share unsolicited pornographic content which its consumers may find offensive or disturbing. In recognition of this, the report attempts to exploit deep learning to demonstrate how a possible solution to this problem may be deployed. In order to do this, a chat application that is capable of identifying and masking pornographic images is developed. In doing so, various prominent deep learning models namely Inception v3, MobileNet v1 and MobileNet v2 are retrained using a pornographic dataset by the application of transfer learning to classify pornographic and non-pornographic content. These models are then evaluated using different model deployment paradigms (on-device and cloud-hosted). The results of the evaluation along with pertinent literature are then used as a basis to make vital architectural decisions in relation to the development of the aforementioned chat application. The cloud-hosted Inception v3 was deemed the most ideal among the explored model and deployment paradigm combinations.

Keywords: Deep Learning, Convolution Neural Networks, Pornographic Images on Social Media, Classification of Adult Images, Inception v3, MobileNet, iOS Application

Declaration

We certify that:

- This thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university, and that to the best of our knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

- Where necessary we have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department.

- The thesis is **9526** words in length (excluding text in images, tables, bibliographies, and appendices).

Acknowledgement

We would like to extend our gratitude to Professor Richard Sinnott for providing us with the opportunity to pursue this project and for his constant support throughout its duration. We would also like to thank Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle and Arnaldo de A. Araújo for providing us access to the NPDI dataset that was used as part of this project.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The advancements in technology over the past two decades, especially in the field of networking and hardware have led to a massive proliferation in the number of internet and smartphone users. These numbers continue to grow at a staggering rate. As a result, applications that allow users to share content as a means to facilitate social interactions have begun to gain increased prominence [1]. The most salient among them being chat applications and social networking platforms. Due to the inherent nature of these applications, their prevalence in contemporary society has led to numerous ramifications that range from technical and legal issues relating to security and ownership of data all the way to more philosophical concerns pertaining to the moral and ethical implications of such applications on society as a whole. One such issue that has been identified to be rampant in such applications stems from the ability of users to share unsolicited pornographic content which may be construed by its consumers as being lewd in nature [2]. Hence, there is a dire need for a mechanism that can effectively filter out such content so that appropriate steps can be subsequently taken. In recognition of this, the report explores deep learning techniques for the classification of pornographic images and attempts to demonstrate its application in systems that enable content sharing for the purposes of social interaction. In doing so, the report documents the development and working of a real-time chat application that is capable of classifying and masking pornographic images as a way of demonstrating how a pragmatic solution to this problem could be potentially deployed.

1.1. Background

The sharing of images using social applications[1] has become a pervasive practice in modern society. As of 2019, the instant messaging application WhatsApp reports that a venerable 4.5 billion images are sent using the application every day [3]. On the other hand, Snapchat another popular multimedia chat application reports that out of the 3 billion snaps (videos and images) that are shared every day using the platform, more than 56% of it takes the form of images [4]. Thus, it could be tenably stated that images have become one of the most dominant media for content sharing in social applications. This upsurge in image sharing has led to concerns relating to the appropriateness of the content within these images which has in turn compelled administrators of numerous social applications to implement various content moderation mechanisms to ensure that the users adhere to the community guidelines of these social applications. The majority of these mechanisms are based on the reporting or flagging of content by the application's users which are subsequently investigated by content moderators (generally humans) [5][6]. A major problem with this approach is that it attempts to handle issues after its occurrence which might not be the best approach when the issue is related to the dissemination of pornographic content. A proactive technique that can identify pornographic content and take appropriate actions might be a more appropriate mechanism in this context. Additionally, another disadvantage of the current moderation approach is the human intervention (involvement of content moderators) that makes the process expensive and time-consuming. An alternate approach would be to use image analysis to identify pornographic content so that the required steps can be taken before it is shared with

[1] Social applications/Social systems: An application that allow the sharing of content to facilitate social interactions

other users of the system. This approach could mitigate these identified limitations that are inherent in current content moderation subsystems.

1.2 Purpose of the Project

The project had two main objectives which are as follows:

i. To explore deep learning for the classification of adult images by evaluating the performance of prominent convolutional neural networks hosted using different deployment paradigms (on-device and cloud-hosted).

ii. To demonstrate a potential solution to the unsolicited pornographic image dissemination problem by the development of a real-time chat application that is capable of identifying and masking pornographic images.

1.3 Project Overview

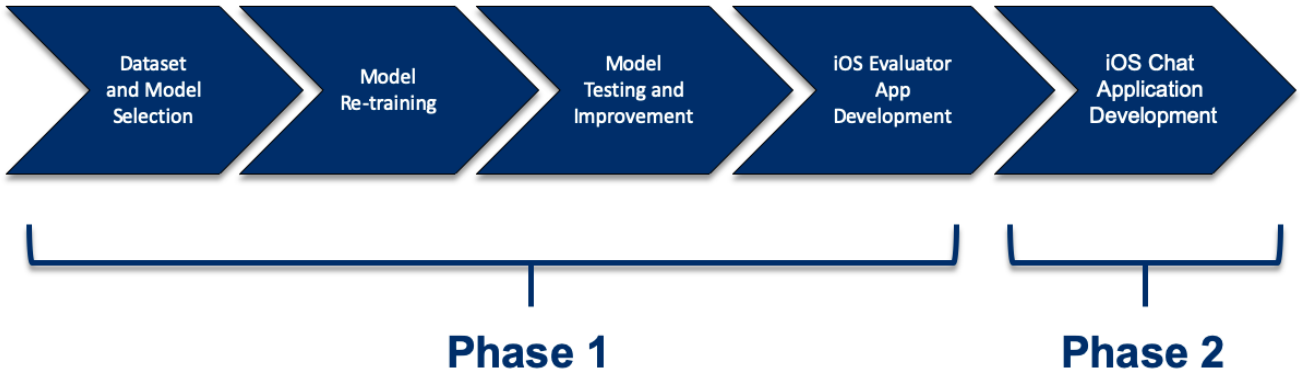


Figure 1.3: Project Overview

On the basis of the proposed objectives, the project was divided into two phases:

Phase 1:

As part of Phase 1 of this project, various prominent deep learning models namely Inception v3, MobileNet v1 and MobileNet v2 were retrained using a pornographic dataset by the application of transfer learning to classify pornographic and non-pornographic content. Subsequently, a mobile application was developed to deploy the retrained models using different deployment paradigms. These models and deployment paradigms were then evaluated using various performance metrics.

Phase 2:

The second phase of the project dealt with the development of a real-time chat application that is capable of identifying and masking pornographic images. The results of the evaluation conducted as part of Phase 1 along with pertinent literature were used as a basis to make vital architectural decisions in relation to the development of this application.

Chapter 2: Related Work

A majority of the recent work that dealt with the identification of pornographic content in still images used image analysis techniques that exploit machine intelligence in some capacity. Intelligent processing techniques usually involve the extraction of certain preselected features from the image that needs to be classified. Methods that made use of traditional machine learning techniques generally entailed a human dominant feature selection process which was often based upon domain-specific knowledge pertaining to the classification problem at hand. The most extensively investigated feature for the purposes of identification of pornography was deemed to be the colour of human skin [7][8]. Additionally, Support Vector Machine (SVM) was found to be one of the most commonly used classification models for the identification of pornographic content in images [9].

Several studies that use human skin colour for the identification of pornographic content have been put forward [7][8][9]. Techniques that use the amount of human skin in images as a high weighted feature are based on the rationale that dictates that if an image contains too many pixels that possess the colour that falls within a range that is selected to represent the gamut of the human skin colour, then this could be taken as an indicator of nudity. This conjecture may not always be valid as it is quite plausible that images that are non-pornographic in nature may contain a high number of pixels whose colours fall within the selected gamut of the human skin colour, this may include images of people partaking in innocuous activities where a high skin expose is quite common such as sumo wrestling, suntanning etc. Hence, skin colour alone is not a reliable factor for the inference of pornographic content in images.

In order to improve upon the aforementioned technique, other features in addition to skin colour were considered in numerous studies. This included shape related features. A number of different methods have been proposed to derive these shape related features which include the extraction of the outlines of the skin regions (contours) which are then used as features [10], modelling the human body by imposing geometric constraints based on the human structure [11] and extracting skin distribution-based shape descriptors [12] .

Advancements in the field of computer vision and the introduction of better visual recognition models have made pornographic image detection more accurate. For instance, the visual bag-of-words (BoW) model provided a non-trivial increase in the accuracy of the germane classification task by using pornographic images to create a visual codebook [13].

Deep learning has recently begun to gain traction as a viable solution for the identification of pornographic content using image analysis. Unlike techniques that use traditional machine learning, deep learning-based approaches integrate both feature extraction and classification into one module and enable minimal to no human intervention in the feature selection process. Additionally, approaches that used deep learning were found to be more accurate as compared to the state-of-the-art approaches that used traditional machine learning along with human-based feature selection for the identification of pornographic content [9]. In recognition of these facts, deep learning techniques were selected for the purpose of image classification in this project.

Chapter 3: Background Research

3.1 Deep Learning

Machine learning is widely used in a plethora of modern applications. The concepts of machine learning are employed from image classification and sentiment analysis for social networks to recommendations on e-commerce websites. Machine learning models are used to identify objects, find patterns in raw data, transcribe speech to text, etc. Conventional machine learning methods are limited by their requirement of substantial domain expertise (human expertise) for designing feature extractors to extract relevant features from raw data. These features are used to build a classifier. The process of feature selection by domain experts can be complex and time-consuming. Hence, representational learning methods have begun to gain increased prominence as they can allow the circumvention of human dominant feature selection.

Representation learning is a technique that enables the machine to obtain a feature set from the raw data to build a classifier. Deep-learning techniques are essentially representation-learning techniques with multiple levels of representation. The fundamental characteristic of deep learning is that these layers of features (representation) are not outlined by human experts but are acquired from data using representational learning techniques. In comparison to traditional machine learning techniques, deep learning techniques have portrayed exceptionally better performance in a number of learning tasks including image recognition, speech recognition and prediction of drug molecules [14].

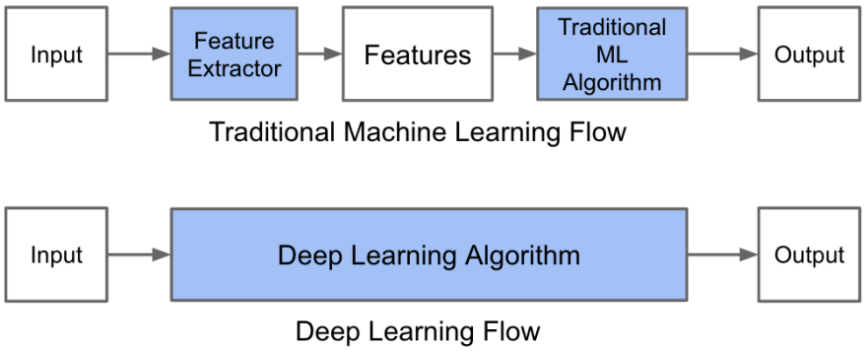


Figure 3.1: Comparison between Deep learning and Machine learning

3.2 Convolutional Neural Network

Convolutional neural networks (ConvNet) is one of the most widely used deep learning architectures in the field of visual imagery analysis [15]. In comparison to traditional deep learning models, ConvNet requires relatively fewer parameters which reduces the learning time, thus making it easier to train [16].

3.2.1 Architecture of Convolutional Neural Network

The ConvNet architecture is inspired by the function and connectivity of "*Neurons"* in the human brain. It consists of an input layer, several hidden layers, and a classification layer. The hidden layers consist of convolution and pooling layers which are used for feature selection. Every convolutional layer is followed by an activation function, which is called a ReLU (Rectified Linear Units) layer.

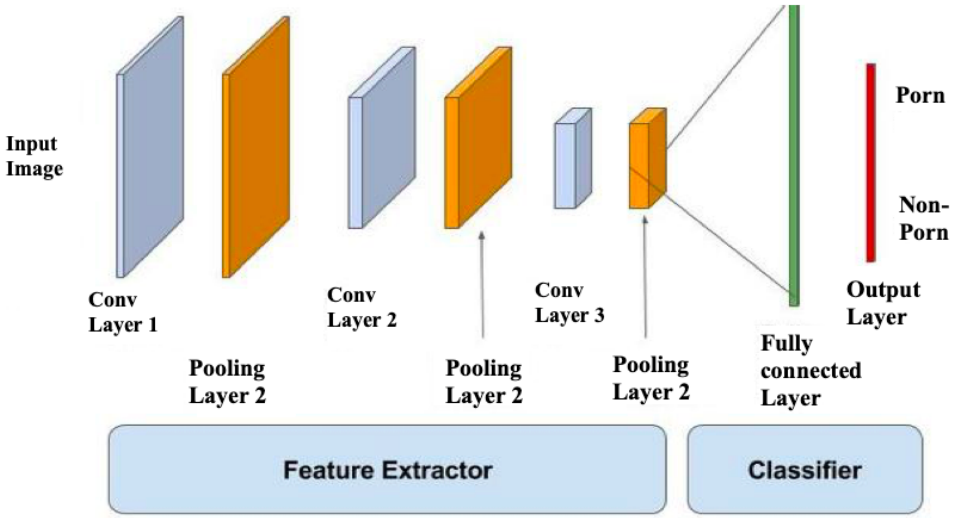


Figure 3.2.1.1: Convolutional Neural Network Architecture

1. Input Layer: The image is fed to the ConvNet system through the input layer. This layer represents images as an array of pixel matrix with dimensions as width, height, and depth, where depth represents the RGB channel [17].

2. Hidden Layer: The hidden layer is made up of convolutional layers, pooling layers, and ReLUs.

a. *Convolutional layer:* The main goal of the Convolutional layer is to extract the features from the input image. A filter or kernel is a small matrix used for feature detection. Each Convolutional layer is connected to the succeeding layer using weighted links. The output of these layers is a feature map that is obtained using a dot product between the layers.

b. *ReLU layer:* Each convolutional layer is followed by a ReLU layer. This layer applies an elementwise activation function f(x) to the output of the convolutional layer which replaces the negative values with zeros. This layer also introduces non-linearity in the system to help mitigate the gradient problem which slows the network from converging.

$$f(x) = \max(0, x) \qquad [18]$$

c. *Pooling layer:* The Pooling Layer is applied soon after the ReLU layer. It reduces the spatial dimensions of the input. The two significant advantages of using the pooling layer are that it subdues the computational complexity of the model and controls overfitting problems.

There are different types of pooling layer options such as Max pooling, Average pooling, and Sum pooling. The most widely used type of pooling is Max pooling (2x2 filter size) [19], which takes only the brightest pixel of the input.

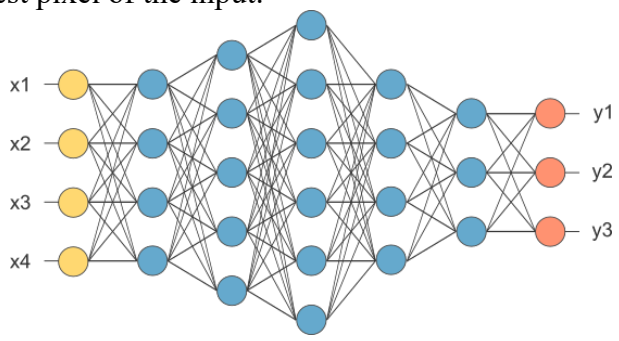


Figure 3.2.1.2: Fully Connected Network [20]

3. Classification Layer: The output from the previous layers is a matrix, which is flattened to produce a vector as shown in figure 3.2.1.2. This vector (x1, x2, x3, and x4) is fed to the fully connected layer, which acts as a classifier. This vector represents the probability that a certain feature belongs to a certain label (y1, y2, and y3). For example, an obscene image may have more skin and ergo the feature representing skin will have a high probability associated with the porn label. Finally, there is an activation function such as sigmoid or softmax, which will determine the label based on these probabilities.

3.2.2 Training Convolutional Neural Networks

Before starting a ConvNet, the filter values or weights at the different layers are randomized. The filters have no information regarding what to look for in an image. In order to converge the network to an optimal solution, labelled data (training data) is provided and the filter weights are adjusted. The network can adjust its weights through a process called backpropagation.

Backpropagation can be divided into four different stages: Forward pass, Loss function, Backward pass, and Weight update. At forward pass, the training image is fed into the input layer. Since the weights are randomized at the beginning, the output is not classified correctly. Nevertheless, the actual output and the predicted output are used for the calculation of the error value. This error value calculation is contingent on the loss function selected for a specific problem type. Cross-entropy loss (also known as log loss) is one of the most commonly used loss functions for classification problems [21].

$$E_{cross-entropy} = -\sum_{i=1}^{N_{output}} (t_i \times \log(y_i) + (1 - t_i) \times \log(1 - y_i))$$

Backward pass is carried out throughout the network in order to find the weights which will significantly reduce the error value produced by the loss function. The decrease in error value implies that the predicted output is converging towards the targeted output/label. The last step is to update the weights. This process carries on for many iterations until the network reaches its optimal state. The new weights are calculated using the error value, the initial weight and the learning rate using the following equation (given by [22]):

$$W = W_i - \eta \frac{dL}{dW}$$

Where,

W = weight
W_i = Initial weight
$L = E_{cross-entropy}$
η = Learning rate

The learning rate is chosen at the beginning of the training process. A higher learning rate would lead to relatively quicker network convergence as the weights are updated by a large value. However, this might cause the model to converge to suboptimal values thus resulting in underfitting. Alternatively, a smaller learning rate increases the number of iterations required for the model to converge. That being said, this may lead to overfitting as the model unknowingly learns too much about the dataset to an extent that it adversely impacts the performance of the model. The learning rate often ranges between 0.0 and 1 [23].

3.3 Transfer Learning

In traditional learning methods, a model is trained on a dataset from scratch with zero knowledge of the data. Eventually, the model aggregates a common pattern across training data until it is able to label legitimate data with higher confidence. This approach is tedious and doesn't allow flexibility towards similar tasks. Transfer learning is the technique of using the knowledge gained by a model (also known as a pre-trained model) from one task and use some or all of that knowledge as the starting point on another related task. The main advantage of using a pre-trained model is that the learned feature maps can be directly applied, thus, reducing the time to train the model on a new dataset.

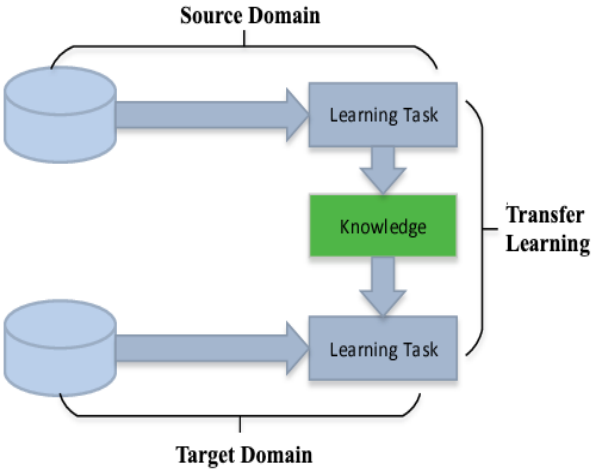


Figure 3.3.1: Transfer Learning

The process of transfer learning can be carried out in three steps: pre-trained model selection, classification of the problem, and fine-tuning.

1. **Selection of pre-trained model**
   This step deals with choosing a suitable pre-trained model out of the wide range of available models. For example, Keras is an open-source neural network library written in Python which provides various models such as VGG 16, ResNet, InceptionV3, MobileNet that can be retrained by applying transfer learning.

2. **Classification of the problem**
   The classification problem can be determined using the matrix shown in figure 3.3.2. This matrix determines which fine-tuning strategies are to be applied depending upon the size of the dataset and its resemblance to the pre-trained model's dataset (discussed later).

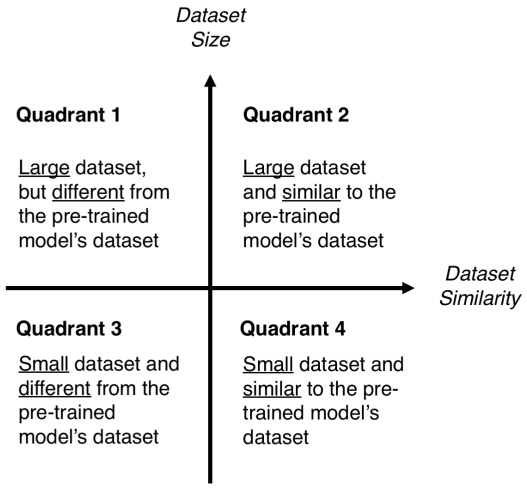


Figure 3.3.2: Classification Problem Evaluation Matrix [24]

3. **Fine-tuning**
   Fine-tuning is the process of introducing small adjustments in the model in order to improve its performance. The three main strategies used in fine-tuning a model are:
   i. Training of the entire model from scratch
   ii. Training only some layers of the model
   iii. Training only the classifier

   Depending on the matrix shown in figure 3.3.2, one of the above fine-tuning strategies can be applied:
   1st Quadrant: If the classification problem falls in 1st quadrant, then the entire model needs to be trained.
   2nd and 3rd Quadrant: If the problem falls in the second or third quadrant, then only some layers of the pre-trained model need to be trained and the rest remains unchanged.
   4th Quadrant: When the datasets are similar and the size of the new dataset is small, then the convolutional layers are frozen, i.e. the output from the pre-trained convolutional layers (Feature extractor part) is fed straight to the new classifier.

3.4 Prominent Convolutional Neural Networks

Some of the most widely used CNN models for image classification are discussed below:

3.4.1 MobileNet

MobileNet v1 was designed for running computer vision applications efficiently on mobile devices. It replaced the computationally expensive traditional convolution layers with depthwise separable convolutions in order to obtain faster performance on mobile devices [25]. MobileNet v1 provides improved user experience, privacy, and security. MobileNet v2 is an updated version of MobileNet v1. It has an improvised depthwise separable convolution block as shown in figure 3.4.1(b) which yields better accuracy, faster performance, and has lesser parameters as compared to MobileNet v1 [26].

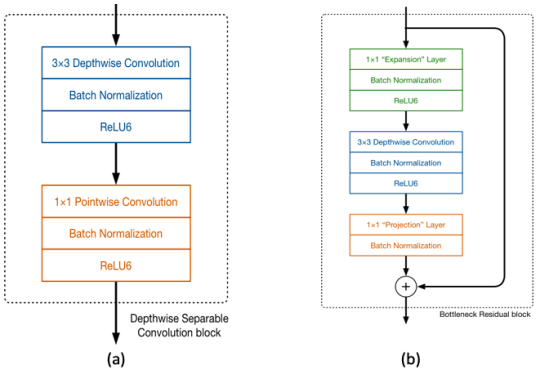


Figure 3.4.1: Depth wise convolution block of (a) MobileNet v1 (b) MobileNet v2 [26]

3.4.2 Inception

Inception was introduced by Google with an intent to reduce the costs associated with computational resources without affecting the accuracy of the model in image classification using deep learning [27]. It is based on the GoogleNet architecture seen in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014. Inception is one of the most widely used image classification models and boasts an accuracy of 78.1% on the ImageNet dataset [28].

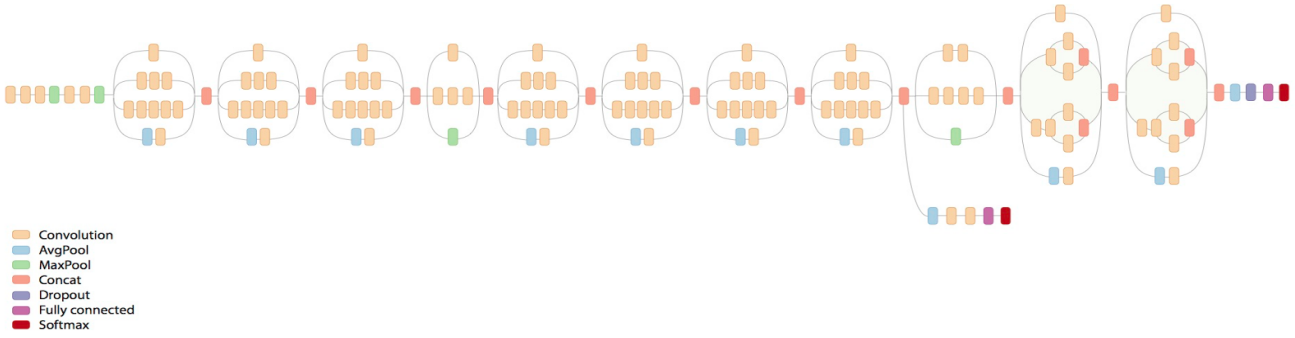


Figure 3.4.2.1: Inception V3 Architecture

The Inception V3 architecture comprises of 11 inception modules as shown in figure 3.4.2.2. Each module is stacked on one top of the other as shown in figure 3.4.2.1. Factorization was implemented in the Inception v3 convolution layer to reduce the computational cost and the total number of parameters which in turn helps in reducing problems associated with overfitting.

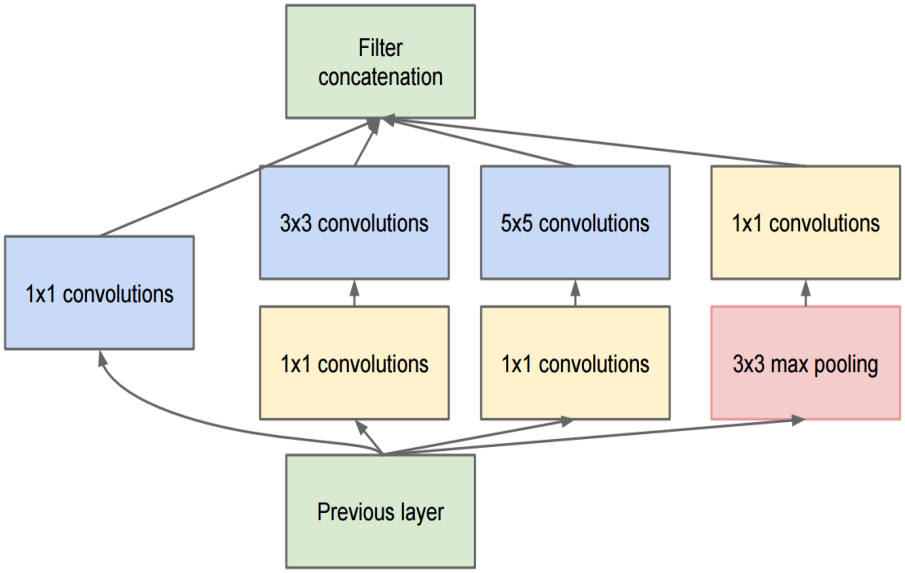


Figure 3.4.2.2: Inception Module

3.4.3 Neural Architecture Search Network (NASNet)

Neural Architecture Search (NAS) is the process of automating the design of Artificial Neural Networks (ANN). The NAS system requires the dataset and the task (regression, classification, etc.) to produce the desired architecture. NAS is employed in developing architectures that perform equal to or better than manually derived ANN architectures. However, this process of designing architectures for a large dataset is computationally expensive. This issue was addressed by NASNet by proposing a concept of transferability, where an architectural block is designed on a small dataset and then transferred to a larger dataset. The novel architecture or NASNet is obtained by adding the layers that performed well on small datasets like CIFAR-10 that used a layer of Normal Cells that returns feature maps of the same dimension and by adding layers that performed well on large datasets like Imagenet made of Reduction Cells that returns feature maps with dimension reduced by a factor of 2 [29]. Figure 3.4.3 shows the NAS architecture for the CIFAR-10 and ImageNet dataset [30].

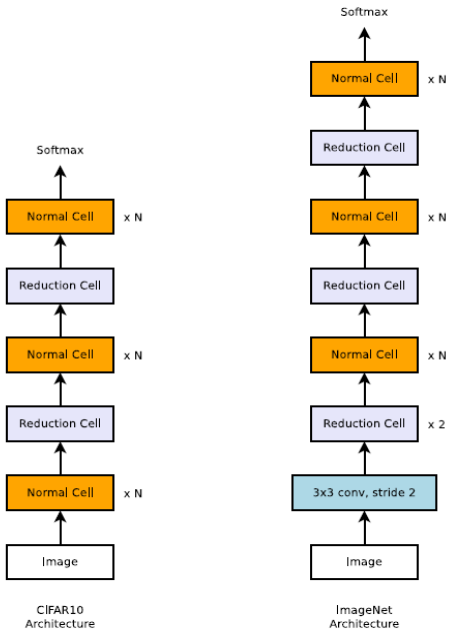


Figure 3.4.3: CIFAR-10 Architecture and ImageNet Architecture

3.5 Model Deployment Paradigms

Deep learning models can be hosted on-device or on the cloud. An appropriate paradigm is selected based upon a few trade-offs (speed, accuracy, etc.). Cloud-hosted models allow computation offloading by allowing applications to interact via API calls, thus allowing the mobile application to stay light and energy-efficient. Additionally, cloud-hosted deep learning models can be more resource intensive as they are not restricted by the device's resource limitations and can, therefore, provide better accuracy. The downside of these models is the significantly higher inference time as compared to the models deployed on-device owing to the time associated with sending and receiving data to and from the model that is deployed on a remote server (on the cloud) [25]. On-device deployment, on the other hand, provides greater data privacy and faster inferences. However, on-device deployment requires the mobile device application to be updated each time any change is made to the model. Instead of compressing the size of a model to make it available on-device, ConvNet models like MobileNets are used which are computationally less expensive, smaller in size and provide more or less accurate results. Additionally, the on-device models can be used without an internet connection as it does not require a request to be sent to any cloud server. The best deployment paradigm often depends on the relevant application use case, the system requirements specification, the architecture of the system and the context in which the application is deployed.

3.6 TensorFlow

TensorFlow is a python-friendly, open-source software library introduced by Google for machine learning applications. TensorFlow provides researchers comprehensive and flexible tools, libraries and resources to build and deploy machine learning applications [31]. TensorFlow was used to train and deploy the deep learning models used in this project.

3.7 CoreML

Core ML is a machine learning framework developed by Apple. It allows the integration of machine learning models in iOS applications. In order to explore different deployment paradigms (on-device as well as cloud-hosted) as part of Phase 1 of this project, CoreML was used to deploy models on the iOS device.

3.8 Firebase

Firebase is a standalone MBaaS (Mobile Backend as a Service) platform owned by Alphabet Inc which has been integrated into the Google Cloud Platform [32]. It provides a multitude of services that aid in the development of mobile and web applications. Firebase is used as an integral part of the back end for the chat application developed as part of phase 2 of this project. It is specifically used to perform user authentication and synchronization of data across multiple application users.

Chapter 4: Retraining of the Deep Learning Models

4.1 Selection of the Dataset

The NPDI dataset is one of the most widely used pornographic datasets for the development of computer vision applications. It was constructed by the NPDI group from the Federal University of Minas Gerais in Brazil. In recognition of its considerable performance in many image processing applications, the NPDI dataset was chosen for training the models in this project.

The NPDI database contains 400 pornographic and non-pornographic videos, which sums up to 80 hours of videos. The dataset is broadly classified into three main categories based on its content: the porn category, the easily distinguishable non-porn category, and the difficult non-porn category [8]. The pornographic category incorporates diverse genres of pornographic material and includes partakers of different ethnicities. The pornographic content was extracted from various pornographic websites, while the difficult non-pornographic content was extracted from the internet using keywords like swimming, suntanning, and wrestling, which are activities where participants often tend to expose more skin. Some examples of the NPDI dataset are shown in figure 4.1.1.

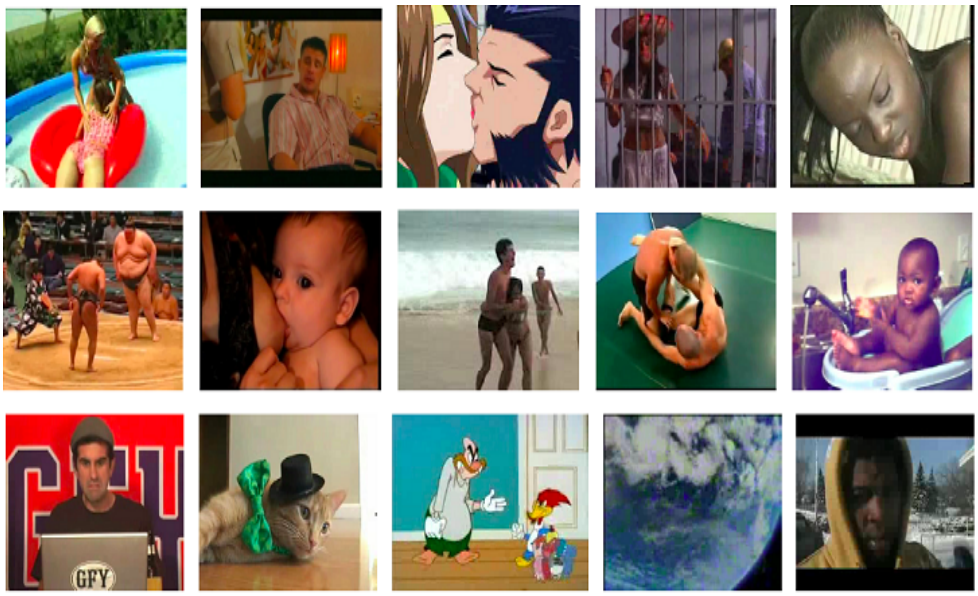


Figure 4.1.1: Samples of NPDI dataset [8]

A total of 16,727 keyframes were extracted from the 800 videos by segmenting these videos into shots. The keyframe is the middle shot of each video. Out of these 16,727 frames, 10,340 were non-pornographic while 6,387 were pornographic in nature.

Table 4.1.1: Contents of the NPDI Dataset

Class	Videos	Hours	Key Frames per Video
Porn	400	57	15.6
Easy Non-porn	200	11.5	33.8
Difficult Non-porn	200	8.5	17.5
Total	800	77	20.6

On evaluating the dataset with respect to the description of pornographic and non-pornographic content as proffered by [33], the NPDI dataset was inferred to be labelled erroneously i.e. some of the non-porn images were found to be labelled as porn. Thus, impelling the need for data cleansing. Some of the examples of wrongly labelled images are shown in figure 4.1.2.

Figure 4.1.2: Erroneously labelled non-porn images

4.2 Data Cleaning

The primary aim of data cleaning is to recognize and remove inaccurately labelled images in order to produce an accurate dataset. Wrongly labelled datasets can largely influence the performance of the deep learning model. Hence, data cleaning is of great significance in order to curb misclassifications. Data cleaning was performed on the NPDI dataset by removing the non-porn images from the porn folder. The number of porn and non-porn frames after performing data cleaning are shown in table 4.2.1.

Table 4.2.1: Number of images after data cleaning

Class	Number of Images
Porn	4,573
Non-porn	10,341

4.3 Data Augmentation

Data augmentation is a procedure that can enhance the performance of neural networks by increasing the diversity of the available data through techniques like image flipping, image rotation, image cropping, de-texturizing, decolorization, Gaussian noise, etc. [8]. The following techniques were selected for this project:

i. **Image Flipping:** It is one of the most commonly used augmentation techniques, that allows the model to remain unbiased towards features that are only available on one side of the dataset.
ii. **Image Rotation:** This technique allows the model to reduce overfitting by learning about the dataset in different rotations. The images were rotated at an angle of 90 degrees to avoid background noises.
iii. **Gaussian Noise:** This technique is useful for preventing overfitting as it allows the model to give equal weights to slightly distorted images. Noises were introduced with a factor of 2 for this dataset.

4.4 Model Selection

The deep learning models explored as part of this project are as follows:

i. Inception v3
ii. MobileNet v1
iii. MobileNet v2

The rationale that led to the selection of these models was based on the contrasting characteristics that these models manifest. While Inception is known for its relatively higher accuracy, MobileNet offers comparatively quicker inferences at reduced accuracy.

4.4.1 Deployment Paradigm Selection

All of the models mentioned in the previous section were deployed on-cloud and a subset of them were deployed on-device as well (hosting Inception v3 on the device was deemed impractical on account of its size). The models deployed on-cloud were trained on (Nectar) cloud instances and are made available by the means of consumable classification services. These services along with the models were deployed on a Nectar cloud instance. The services are exposed using endpoints written in Python using the Django framework. The following models were deployed on-cloud:

i. Inception v3
ii. MobileNet v1
iii. MobileNet v2

The models deployed on-device were initially trained on-cloud (Nectar) using the TensorFlow platform and then later converted to the CoreML model format using the *"tfcoreml"* library to make the models compatible with the iOS platform. The following models were deployed on-device:

i. MobileNet v1
ii. MobileNet v2

4.4.2 Training Process

The training process was divided into two stages. In the first stage, the pre-trained versions of the selected models were downloaded using the *"imagenet"* weights so as to maximize the potential of transfer learning. The output layer of the pre-trained model was selected as the starting point for retraining the models on the prepared (cleaned and augmented) NPDI dataset. In the second stage, the most suitable training process was determined by fine-tuning the training parameters by gradually increasing the training steps for each model. Table 4.4.2.1 details the training parameters used for the models over multiple executions.

Table 4.4.2.1: Model Training Parameters

	Training Steps	**Learning Rate**	**Training Data (%)**	**Validation Data (%)**	**Testing Data (%)**	**Batch Sizes**
1st Execution	10,000	0.01	70	15	15	64
2nd Execution	15,000	0.01	80	10	10	100
3rd Execution	20,000	0.01	80	10	10	100

4.5 Performance Evaluation

The metrics used to evaluate the performance of the selected models are:

i. Training and Validation Accuracy
ii. Cross-entropy Loss
iii. Average confidence
iv. F1-score

Whereas, the metrics used to compare deployment paradigms are:

i. Size of the model
ii. Inference time

These metrics and observations are subsequently evaluated to gain insights that form the basis for selecting the most optimal model and deployment paradigm for the purposes of the social (chat) application that is developed as part of phase two of this project.

4.5.1 Model Evaluation using accuracy and cross-entropy loss

Models trained using the TensorFlow library log data of various metrics at every step interval which can be accessed and visualized using the Tensorboard library. Tensorboard enables tracking of running accuracy (training and validation) and cross-entropy loss (in case of classification problems). This section analyzes the training and validation accuracy along with the cross-entropy loss (training and validation) for all of the selected models.

i. MobileNet v1:

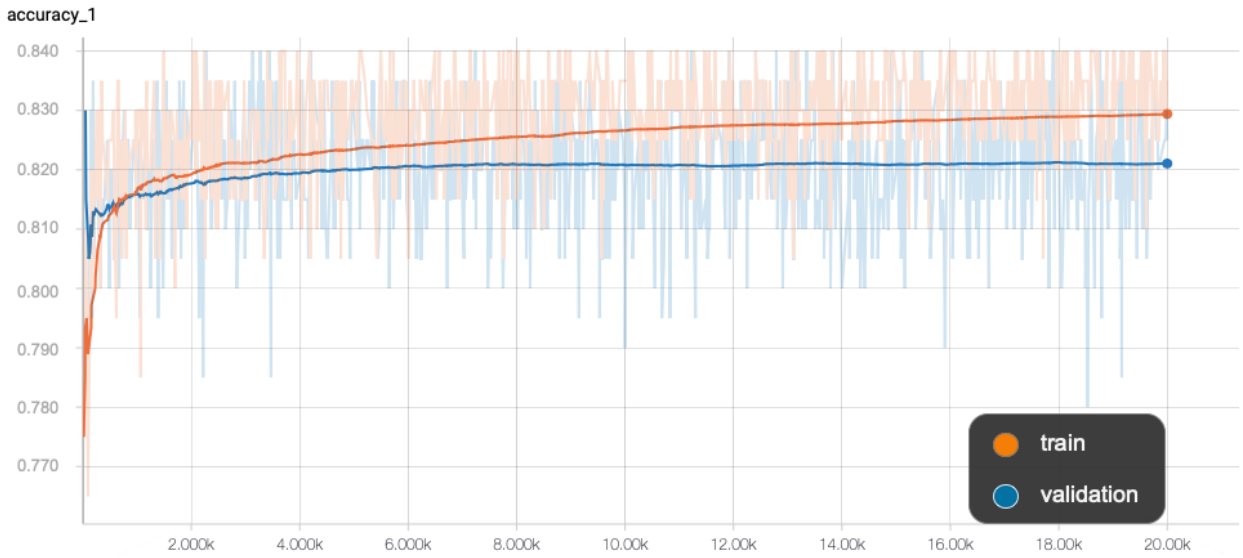


Figure 4.5.1.1: Scalar graph of accuracy for MobileNet v1 - 1

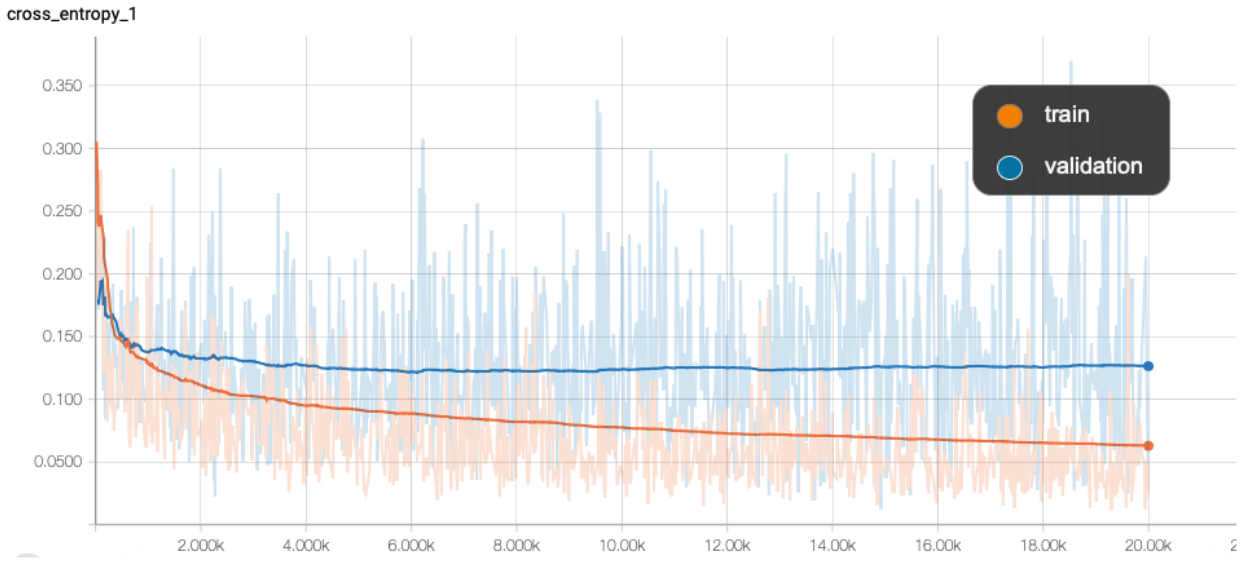


Figure 4.5.1.2: Scalar graph of cross-entropy for MobileNet v1- 2

The validation accuracy for MobileNet v1 becomes stable after 6K steps, however, the training accuracy kept rising until the last step of the iteration. The validation accuracy finally reached

82.1%. Similarly, the cross-entropy loss function became stable after 6K iterations. Among all of the three models, MobileNet v1 had the least accuracy but also the least loss value. Time taken to train the model was ~40 mins.

ii. MobileNet v2:

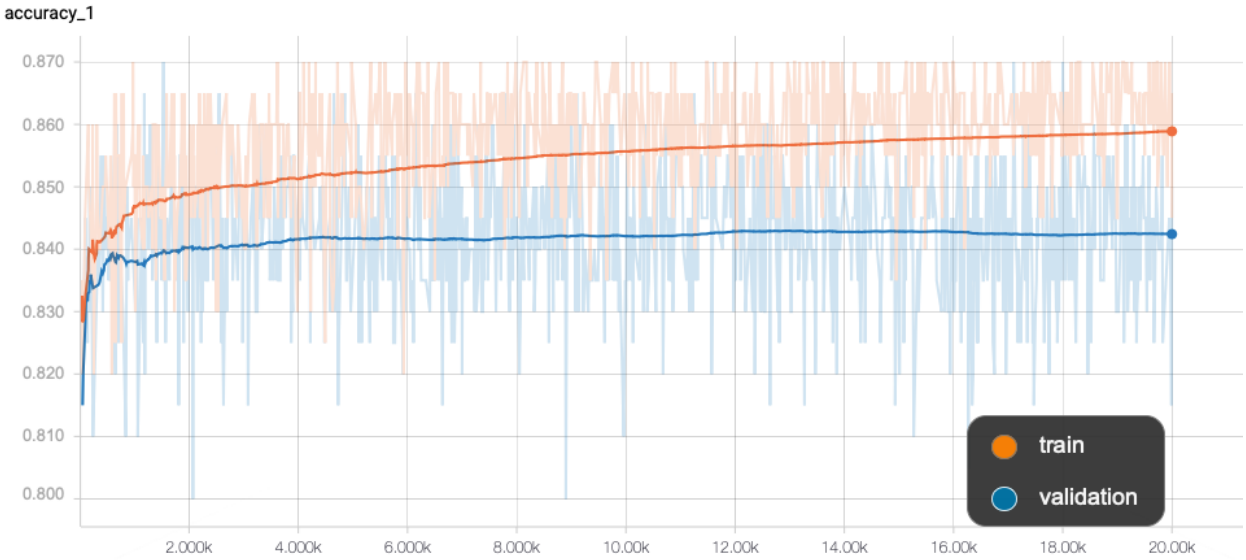


Figure 4.5.1.3: Scalar graph of accuracy for MobileNet v2 - 1

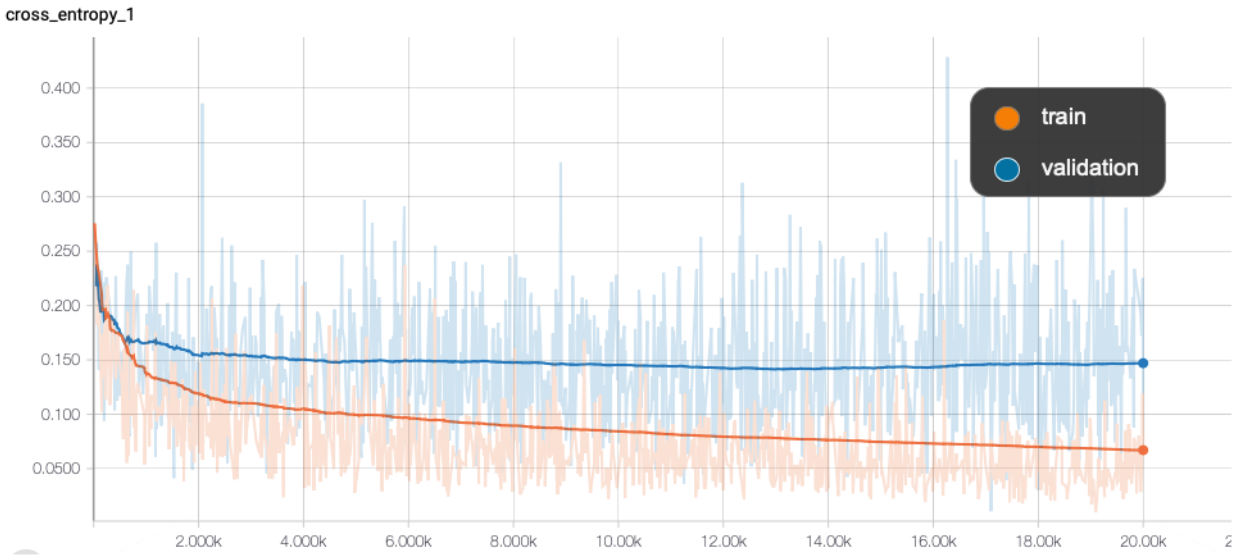


Figure 4.5.1.4: Scalar graph of cross-entropy for MobileNet v2 -2

The validation accuracy and cross-entropy loss for MobileNet v2 became stable after 5K iterations until 16K. Both the accuracy and the loss value slightly increased after 16K iterations. Finally, the validation accuracy reached 84.4%. MobileNet v2 had better validation accuracy as compared to MobileNet v1, but it was heavily penalized by the loss function and hence had the highest loss value among the 3 models. Time taken to train the model was ~48 mins.

iii. Inception v3:

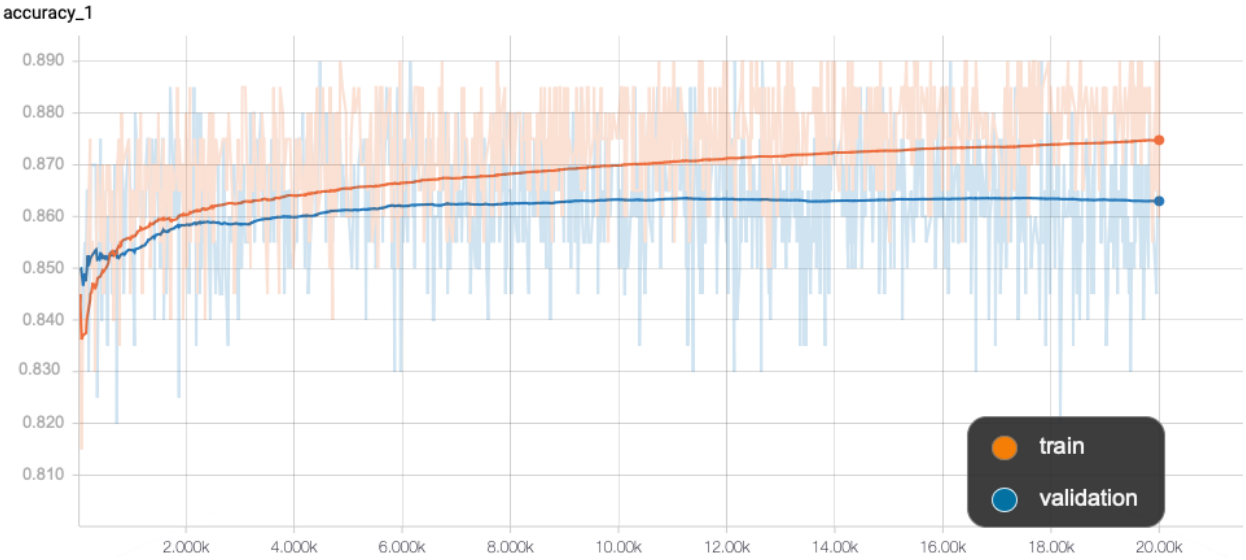


Figure 4.5.1.5: Scalar graph of accuracy for Inception v3 - 1

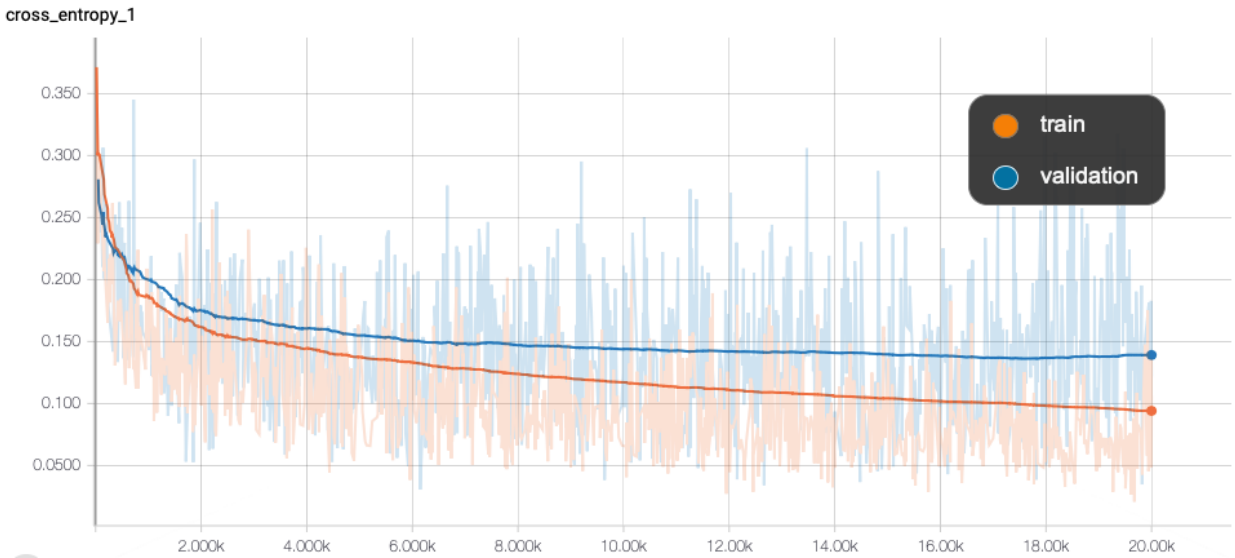


Figure 4.5.1.6: Scalar graph of cross-entropy for Inception v3 - 2

Inception v3 had the highest validation accuracy among all of the three models. However, the training accuracy kept rising until the last iteration. The validation accuracy became stable after 12K iterations. The validation accuracy was 86.4% by the end of 20K iterations. The loss value was also stabilized after 12K iterations and was in between the other two models. Time taken to train the model was ~4 hrs.

4.5.2 Model Performance Evaluation on unseen data

To evaluate the performance of the trained models, 916 images were selected as the testing set. This testing set is downloaded from google using the web crawler library. The testing set is divided into two sub-folders:

1. Non-Porn: This folder includes 558 images of various different categories (e.g. portraits, massage, workout, newborn babies, kissing, etc.)
2. Porn: This folder includes 358 images that are different from the images on which the models are trained.

All of the trained models were tested on the same images, and the results obtained are shown in Table 4.5.2.1.

Table 4.5.2.1: Confusion Matrix for the Selected Models

		Predicted Class		**Models**
		Porn	Non-Porn	
Actual Class	Porn	305	106	MobileNet v1
	Non-Porn	53	452	
	Porn	297	79	MobileNet v2
	Non-Porn	61	479	
	Porn	319	86	Inception v3
	Non-Porn	39	472	

The following metrics were used for the evaluation of the model:

1. **F1-score:** It is the weighted average of Precision and Recall. It is calculated using the formula:

   $$F1\text{-}Score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)}$$

   Where,

   $$Recall = \frac{TP}{(TP + FN)} \quad and \; Precision = \frac{TP}{(TP + FP)}$$

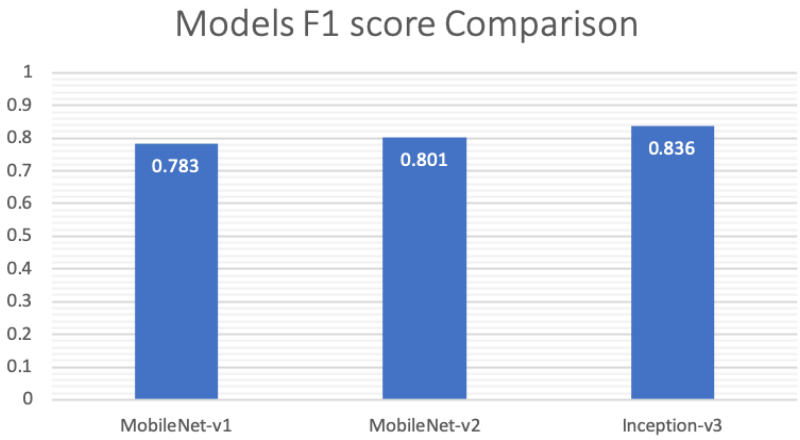


Figure 4.5.2.1: Models F1-score on test set

2. **Testing Accuracy:** It is the ratio of correctly predicted observation over the total number of observations. It can be defined using the formula:

$$Accuracy = \frac{(TP + FP)}{(TP + FP + TN + FN)}$$

Where,

TP - True Positive
TN - True Negative
FP - False Positive
FN - False Negative[2]

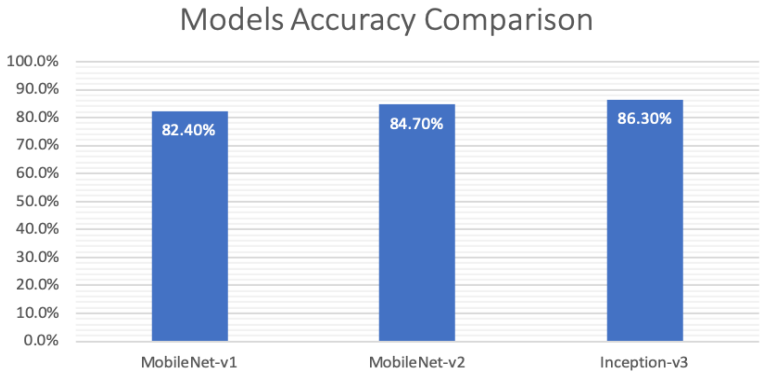


Figure 4.5.2.2: Models Accuracy on the Test Set

Figure 4.5.2.1 and 4.5.2.2 graphically represent the obtained F1-score and accuracy over the test set respectively. It can be inferred from the results that the performance of Inception v3 is better than the other two models.

The accuracy and confidence were also observed for every image in the test set. It was observed that the average confidence for 5 categories in the non-porn group were comparatively low as

[2] Porn class is considered as Positive, while Non-Porn class is Negative

shown in figure 4.5.2.3. The lower average confidence for these categories increased the misclassifications rate for all of the 3 models. This misclassification further affected the accuracy of each model.

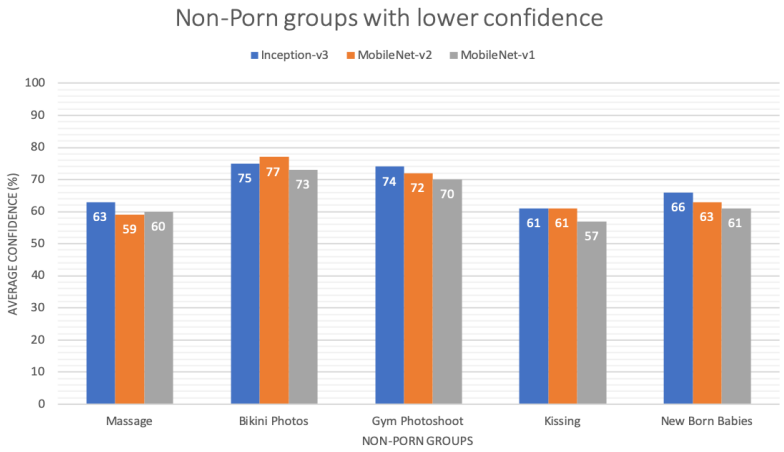


Figure 4.5.2.3: Non-Porn group with lower confidence (avg.)

Following can be the possible reasons for the observed results:

1. Bikini, baby and massage photos tend to have a greater amount of skin exposure which is quite a common occurrence in pornographic images.
2. Gym photoshoots and massage images quite often display facial expressions that are somewhat analogous to that of the partakers within pornographic images.
3. The positions of the human bodies within massage, gym and kissing images are quite similar to that of pornographic images.

4.5.3 Deployment Paradigm Evaluation

This section discusses the different deployment paradigms that were implemented as part of the first phase of this project and the metrics used to evaluate them. Table 4.5.3.1 compares the size of the model and the inference time taken by each model when hosted on-device and on-cloud.

Table 4.5.3.2: Models size and average inference time (on-cloud and on-device)

Models	**Size**	**Avg. Inference Time (ms) on-cloud**	**Avg. Inference Time (ms) on-device**
MobileNet v1	17.1 MB	896	465
MobileNet v2	14.2 MB	917	433
Inception v3	87.4 MB	1743	-

From the results shown above the following observation can be drawn:

1. The models deployed on-device had comparatively lesser inference time.
2. There is a trade-off between accuracy and inference time.

Chapter 5: iOS Evaluator Application Development

5.1 System Overview

The software developed as part of the culmination of phase 1 of the project is an iOS application that is capable of identifying pornographic content in still images, as well as live video streams shoot using the device's camera. This identification is done through image classification that can be done using multiple models as well as different deployment paradigms (on-device as well as cloud-hosted). The application was developed with the intent to evaluate these models and deployment paradigms and ergo is aptly named the "Evaluator".

5.2 System Architecture

The iOS Application

The application is written in Swift 4 and is compatible with iOS11 and above. On-device models are integrated using Apple's CoreML framework while cloud-hosted models are used by consuming an image classification service. The models used are Inception v3, MobileNet v1, and MobilNet v2. However, Inception v3 was not deployed on-device as its large size made it impractical for the purposes of this application.

The Image Classification Service

An image classification service is rendered to the iOS application for invoking off-device models. The service along with the models are deployed on a Nectar cloud instance. These services are exposed using endpoints written in Python by leveraging the Django framework.

5.3 System Functionality

5.3.1 Classifying still images

The application can classify images as pornographic or non-pornographic and present the users with the required results. To select an image, the button titled "Select Image" must be clicked. These images can be selected either from the device's photo library or can be taken from the device's camera by selecting the image source from the pop-over that appears. Once an image is selected, that image would be displayed within the image view area of the application's main screen. To classify the selected image, the application users can simply click the button titled "Classify Image". Subsequently, the application would provide the users with a list of model and deployment paradigm combinations to choose from. Once a model and deployment paradigm combination is selected, a request for classification is sent. For every classification request, the following information is displayed:

i. The status of the classification request.
ii. The confidence by which the image was inferred as being pornographic.
iii. The confidence by which the image was inferred as being non-pornographic.
iv. The inference time.

Note: ii, iii and iv are only displayed once the status of the request is set to “Completed”. Figure 5.3.1 describes the workflow for classifying a still image.

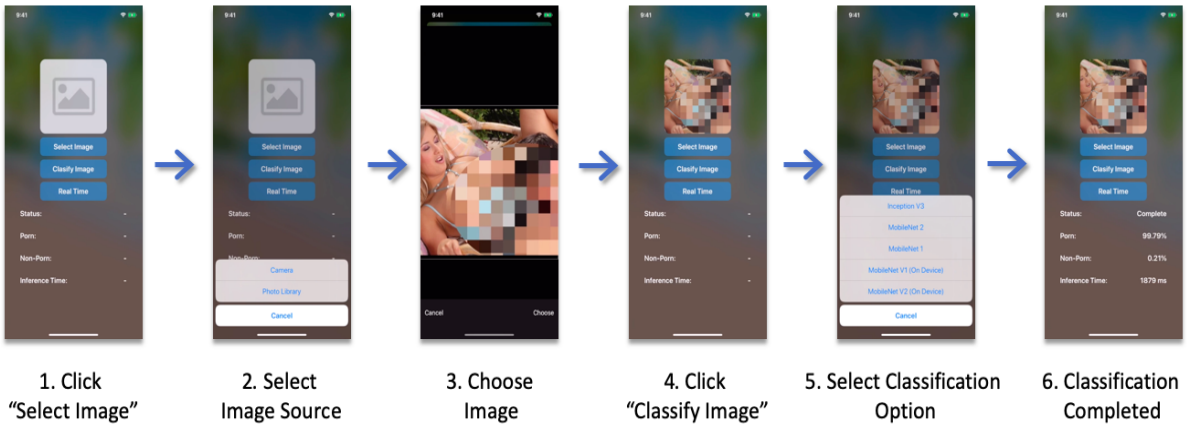


Figure 5.3.1: Workflow: Classifying still images

5.3.2 Classification in a live video stream

The application is also capable of identifying pornographic content in a live video stream taken from the device’s camera. This is done by invoking the selected on-device model on every extracted frame of the video stream. To use this feature, the user can click the button titled “Real Time” which would present the user with the list of deployed on-device models that can be chosen for the classification. On selecting the model, the users would be presented with a screen that displays the live video feed shoot using the device’s camera along with the confidence associated with each class. Figure 5.3.2 illustrates the aforementioned workflow.

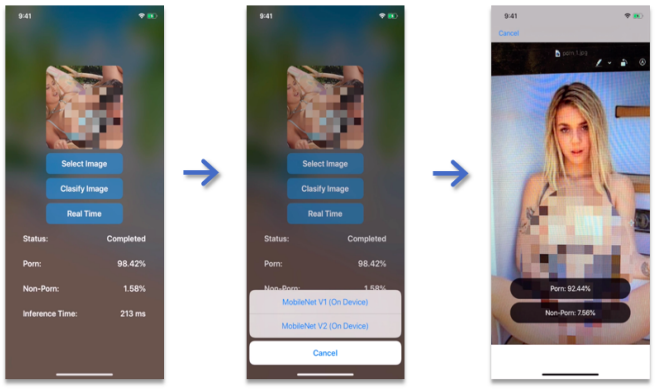


Figure 5.3.2: Workflow: Classification in a live video stream

Chapter 6: iOS Chat Application Development

6.1 System Overview

The software developed as part of phase two of this project is an iOS chat application that provides authenticated users with the capability to connect with other nearby users and send them text and image messages while identifying and masking any pornographic images in real-time. Additionally, the application also provides its end users and administrators with other supplementary features and capabilities that have become commonplace in contemporary chat applications (discussed comprehensively in Section 6.3).

6.2 System Architecture

The system as a whole is made up of three main components namely the application's front-end, the google firebase backend and the Image Classification and Storage Service deployed on the Nectar cloud infrastructure (the latter two together making up the application's backend). Figure 6.2 provides the system's architecture diagram that illustrates the various components of the system and how they interact with each other.

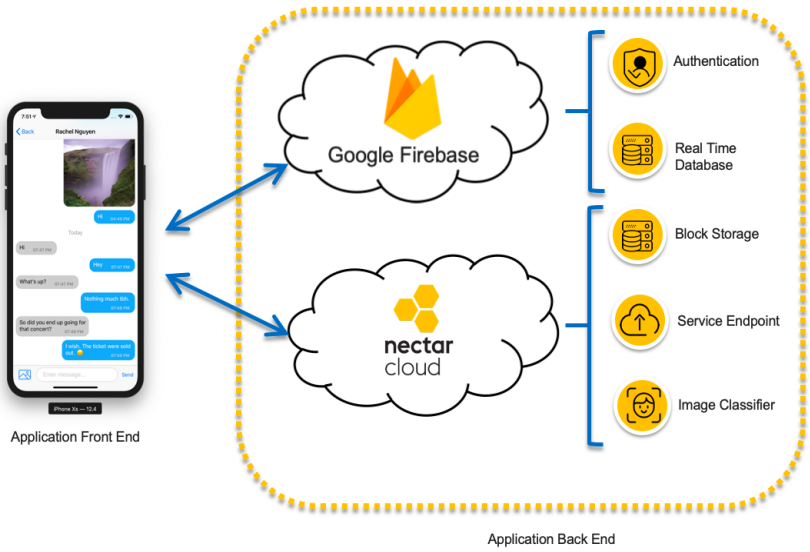


Figure 6.2: Chat Application Architecture Diagram

6.2.1 Application Front End

The application front-end is written in Swift 4 and is compatible with iOS11 and above. It is responsible for providing the application's users with an interface that they can exploit to use and interact with the application. Additionally, it is responsible for interacting with the application's backend to store, manipulate and retrieve relevant data.

6.2.2 Firebase Backend

The firebase component is an integral subset of the application's backend. The system uses Firebase for authenticating users as well as for storing data and synchronizing it across multiple clients in real-time. User authentication is done using Firebase Authentication. Additionally, any data that requires to be managed in order to facilitate the functions of the application are stored in the firebase Real-time database with the exception of the images that are stored using the storage service discussed in the next subsection.

6.2.3 Image Classification and Storage Service

The component as the name suggests renders image classification and storage services to the application's front end. These services are exposed using endpoints written in Python by leveraging the Django framework. The application's frontend can consequently avail to these services by consuming the exposed APIs. These services are deployed on a Nectar cloud instance. The services that this component renders are of two types:

i. **Pure storage service:**
   The pure storage service allows the storage and subsequent retrieval of data objects (images or otherwise). Consumers of this service would typically send an HTTP request along with the data objects to be stored and the service would then store this data and respond to the consumer with a JSON object that contains a URL that could be used to subsequently access the stored data object.
ii. **Image classification and storage service:**
   The image classification and storage service is similar to the pure storage service. However, unlike the pure storage service, this service performs the classification of the images as pornographic or not using the Inception V3 classifier hosted on the same cloud instance as this service. This classification is done prior to storage. In addition to the URL for accessing the stored object, this service's response JSON object would also contain the class confidence.

6.3 System Elucidations

6.3.1 User Authentication

As previously stated, Firebase Authentication is used for user authentication in this application. When users register with the application, their credentials are stored within Firebase authentication and are subsequently used for authenticating the users when attempting to log into their accounts.

6.3.2 Finding Nearby Users

The application provides the users the capability to initiate or continue conversations with nearby users (i.e. users that are in close spatial proximity to the local user). It does this by keeping track of the user's location and updating it as the user's location changes. The user's location is only retrieved and updated when the user moves one kilometer away from its last updated location. This helps prevent unnecessary battery consumption resulting from the constant retrieval and updating of the user's GPS coordinates which renders to be somewhat of an overkill for the requirements stipula0ted by the given use case. Thus, possessing the location of the users, the application is

capable of retrieving users that are close to the local users. This is done by running geo queries against the stored user locations.

6.3.3 Database Design

All application data (with the exception of images) is stored within Firebase's Realtime Database (FRD) which is a NoSQL cloud database used for the storage and real-time synchronization of data [34]. The FRD uses a JSON like tree representation to store data. The FRD stores user profile information, user locations and information relating to the messages that are exchanged between application's users. The nodes at Level 1 in the tree structure and the data that they store are as follows:

i. **users:** The "users" node is a node that stores the user's profile information (all of the user's data except for the user's authentication credentials and the user's location). Every child node of the "users" nodes represents a user profile and uses the user's identifier retrieved from Firebase authentication as the node's key. Each of these child nodes store the following information:
    a. the user's name
    b. the user's email
    c. the URL to be used for the retrieval of the user's profile image

ii. **messages:** The "messages" node stores data related to the messages that are sent between the application's users. Every child node of the "messages" node represents a message and uses a string that uniquely identifies that message as the node's key. These child nodes store the following information:
    a. The message sender's identifier
    b. The message recipient's identifier
    c. The timestamp associated with the message
    d. The message text (only present in text messages)
    e. The message image URL (only present in image messages)
    f. Other image metadata such as the height, width and classification confidences associated with the porn and non-porn class (only present in image messages)

iii. **user_locations:** The user_locations nodes store the location of the users and these values are updated as the user's location changes. This node contains a child node for every user of the application and uses the user's identifier retrieved from Firebase authentication as the node's key. Each of these child nodes stores the location information (the last know coordinates) of the user it represents.

iv. **user-messages:** The "user-messages" node stores information about the messages that every user is allowed to access. While this information can be derived by accessing and combining information stored within the "users" node and the messages "node", the user-message node is maintained to facilitate a fan-out strategy that entails the storage of duplicate data to eliminates slow joins and increases read performance [35]. This is a common mechanism used in messaging applications to ensure high performance.

6.3.4 Usage of the Cloud Hosted Inception v3 Model

The application uses a cloud-hosted Inception v3 for the classification of images. While on-device hosting of deep learning models is irrefutably one of the most compelling options for many applications use cases, a cloud-hosted model was deemed to be a relatively better deployment paradigm for the purposes of this application. Two of the most conspicuous advantages that on-device hosting brings to the table is that it does not require any network connection and it provides a relatively quicker inference time as the images do not need to be sent to a remote server for processing. However, all social application requires some form of network connection and all shared data in a quintessential social application would generally end up on some server, rendering these advantages irrelevant in this context. Thus, using a "process where you store" approach for image classification that uses cloud-hosted models can provide more efficacious results as this approach can leverage significantly greater storage and computing resources which is in contrast to the on-device classification approach that is limited by the resources that the device possesses.

The retrained Inception v3 that was developed and tested in phase 1 of this project was selected for performing image classification in this application as it provided the best overall accuracy and F1-score. While Inceptionv3 is significantly greater in size and requires greater computing resources that the other explored models, these resources can be easily provided owing to the usage of the aforementioned deployment approach. It is worth noting that Inception v3 has a comparatively greater inference time. That being said higher accuracy was preferred as frequent misclassification could provoke vexatiousness in application users and since empirical observation made it apparent that this higher inference time did not noticeably affect the real-time nature of the application.

6.3.5 Hybrid Backend Architecture

The application was originally developed using an architecture that purely used Firebase as its backend (as shown in Figure 6.3.5). In addition to using Firebase Authentication and Realtime Database as described in subsection 6.2.2, the application used Google Cloud Functions for invoking an image classification model hosted on MLKit and for subsequently storing the image in a Firebase Storage bucket. Since Clouds Functions is not available in the australia-southeast1 location (the only Australian location), the firebase project had to be created in the us-east1 location (South Carolina, USA) [36]. However, data storage and classification related tasks were observed to be quite time-consuming rendering it impractical given the real-time requirement of the application. This observation could be attributed to the latency owing to the project location. In order to curtail this latency, an image classification and storage service was developed and deployed on a Nectar cloud instance in an availability zone in Australia. As a result, this initial architecture was transformed into the one described in Section 6.2.

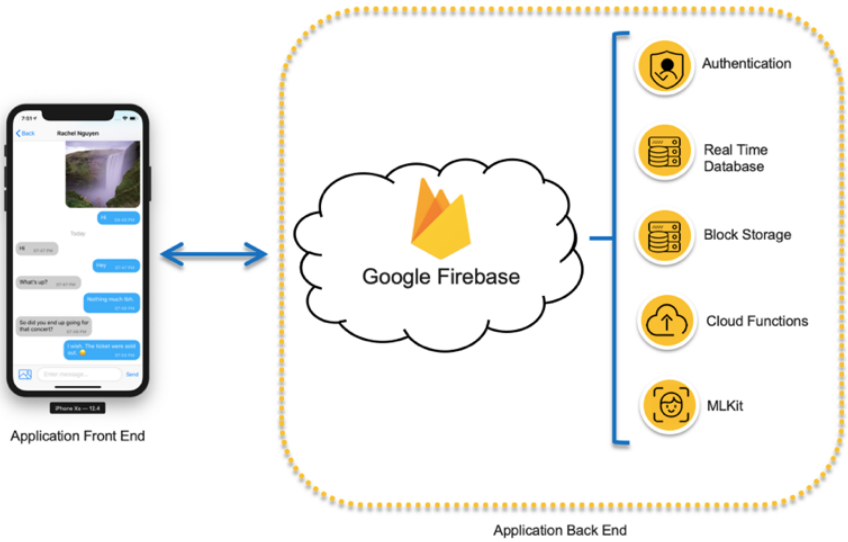


Figure 6.3.5: Initial Chat Application Architecture Diagram

6.4 System Functionality

6.4.1 User Registration

All application users need to be authenticated in order to use the various features that the application provides. Hence, new users need to sign up for a user account by providing the application with the required details so that they can be subsequently authenticated. If a user is not authenticated at the time of the application launch, they are presented with a screen where they can sign up for a new account or login with an existing account. When presented with this screen, new users can register for an account by providing the prompted information and clicking the register button. Additionally, users can optionally select a profile image prior to clicking the register button by clicking on the avatar displayed on the screen which would enable them to select a photo from the device's photo library. Figure 6.4.1.1. illustrates the workflow of a typical registration when valid information is provided. In the cases, where the information entered is invalid, an appropriate message is displayed to the user to notify them of the same. Figure 6.4.1.2 shows some cases where invalid information was provided.

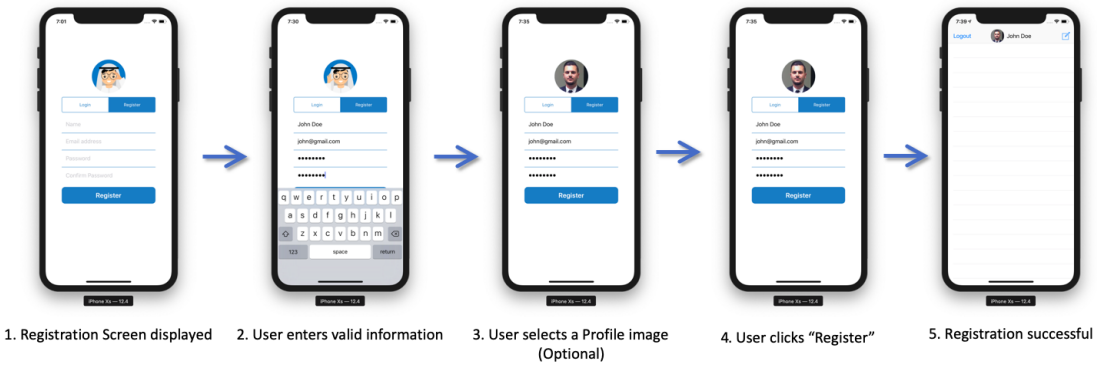


Figure 6.4.1.1: Successful Registration Workflow

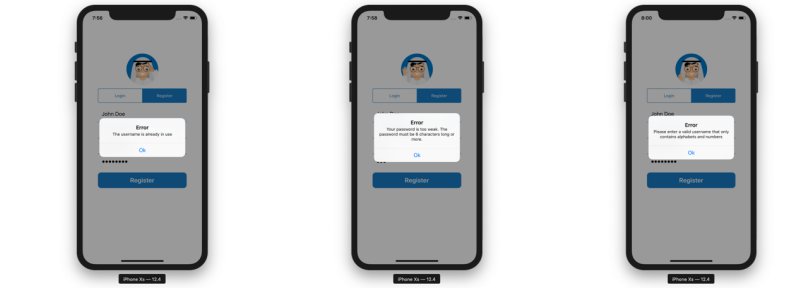


Figure 6.4.1.2: Registering using invalid data

6.4.2 User Authentication (Login)

Existing users can login to their accounts by using their login credentials when presented with the login and registration page. Users can login by clicking the login tab, filling in their credentials and then clicking the "Login" button. Figure 6.4.2.1 illustrates the workflow of a typical login when valid credentials are provided. In the cases, where the provided credentials are invalid or if the user account associated with the entered credentials is blocked, an appropriate message is displayed to notify the user of the same. Figure 6.4.1.2 shows some cases where invalid or blocked account credentials were provided.

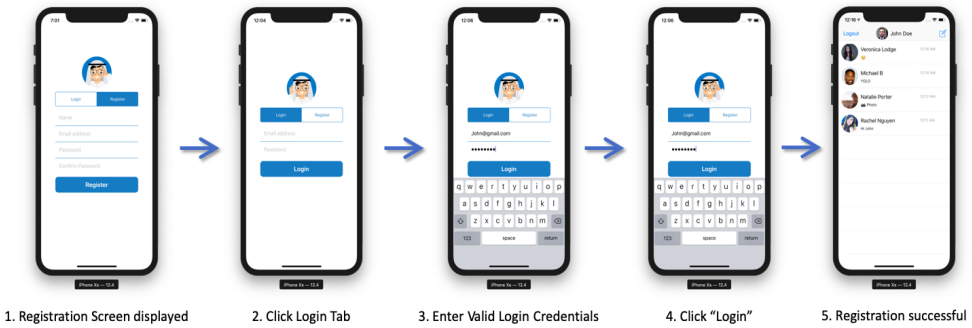


Figure 6.4.2.1: Successful Login Workflow

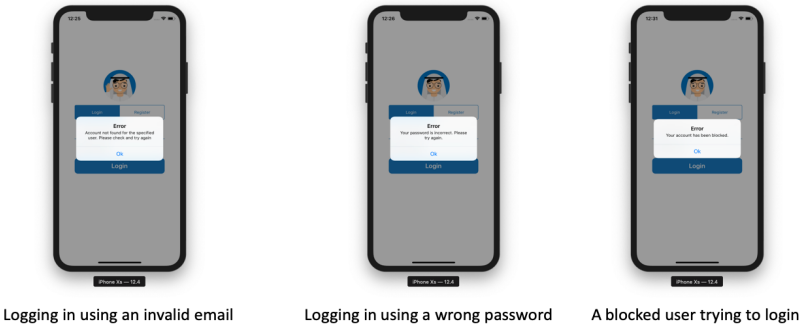


Figure 6.4.2.2: Logging in using invalid or blocked credentials

6.4.3 Listing Nearby Users to Chat With

The application allows authenticated users to view a list of other nearby application users that they can chat with. Once authenticated (after logging in or signing up), the user is directed to the application's main screen (as shown in figure 6.4.3.1). The user can then click on the top right button on the navigation bar of the screen which would present them with a screen that lists all the nearby users that they can chat with along with their respective distances from the local user. Figure 6.4.3.2 shows the germane workflow. By default, the users that are within a hundred kilometer radius of the local user are retrieved.

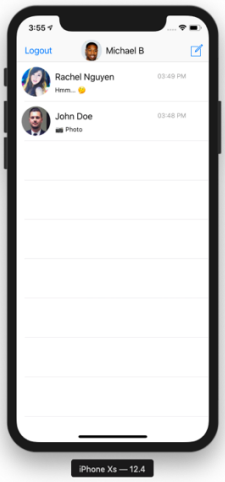


Figure 6.4.3.1: The App's Main Screen

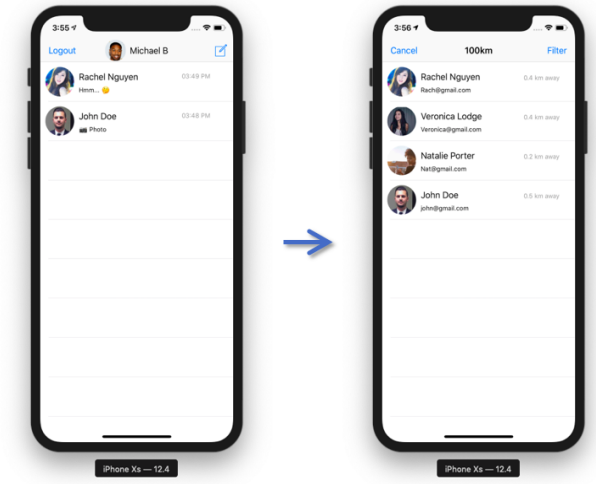


Figure 6.4.3.2: Workflow: Nearby Users

6.4.4 Filtering Users Based on Their Proximity

As mentioned in the previous section, the application enables the user to view a list of other nearby users that they can chat with. While the application retrieves and lists the users that are within a hundred-kilometer radius by default, this radius can be adjusted to filter out users based their distances from the local users. Given the screen that lists nearby users (shown in the final state of figure 6.4.3.1), the local user can click the top right button within the navigation bar titled "Filter". This would bring up a popover that allows the users to change the radius. Figure 6.4.3.1 illustrates this workflow. For example: If a user just wants to see the see the users that are at a distance of two kilometers of less, they would adjust the slider to set the radius to two kilometers and subsequently click "Done". Consequently, the displayed list would reflect the newly selected radius which is also indicated in the title of the navigation bar as shown in Figure 6.4.4.1.

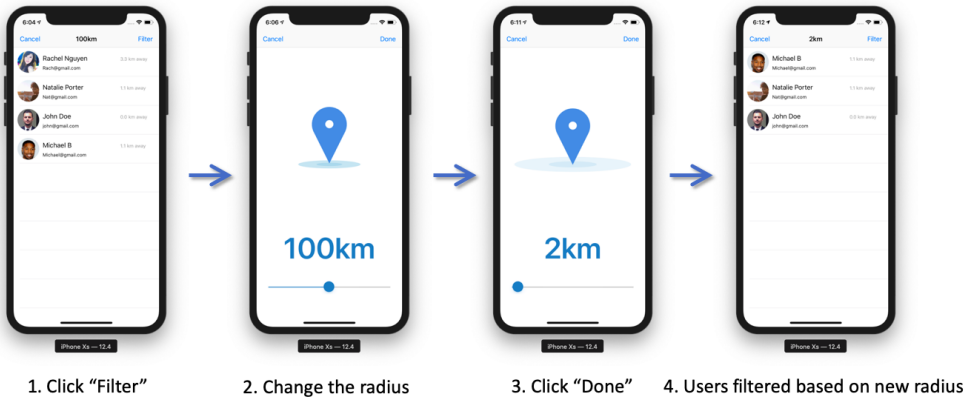


Figure 6.4.4.1: Workflow: Filter users based on proximity

6.4.5 Selecting a User to Chat With

The local user can select users to chat with in two different ways:

6.4.5.1 Selecting a user from the main screen

The local user can select a user to chat with from the application’s main menu which lists the users that the local users has had a conversation with sometime in the past along with last message exchanged between the listed users and the local user along with the message’s respective timestamp. This list is sorted based on these displayed timestamps in chronological descending order and is updated every time a message is sent or received by the local users. To select a user to chat with from the list displayed on the main screen, the users can simply click on that user. This would lead to the presentation of a chat room screen. This screen renders a chat log between the local user and the selected user and set of input controls pinned to the bottom on the screen below the chat log that enables the local user to send messages to the selected user. Figure 6.4.5.1 shows this workflow of selecting a user from the main screen.

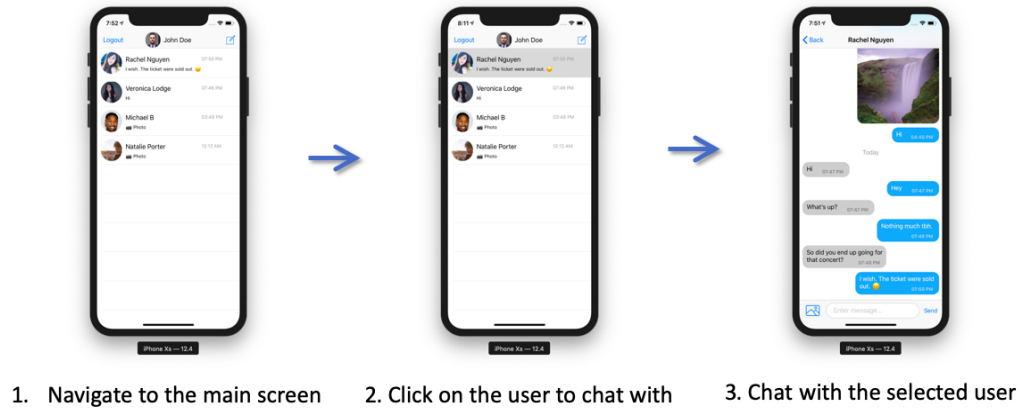
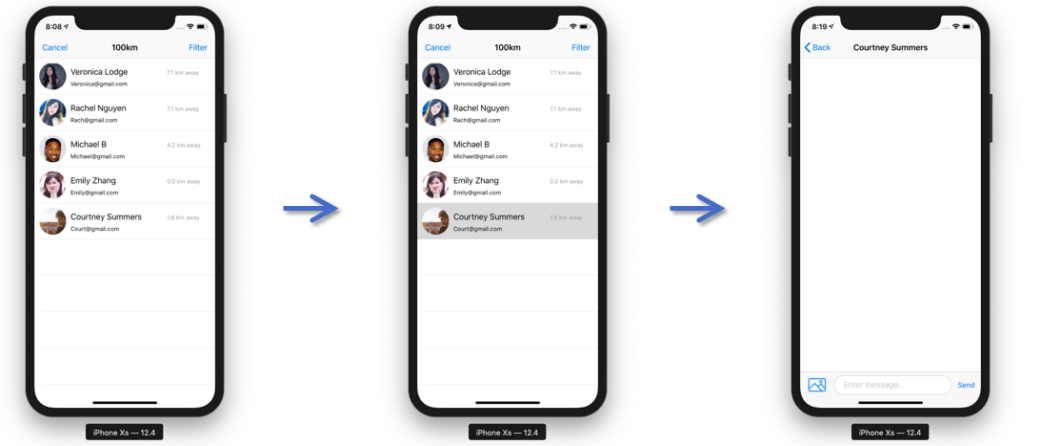


Figure 6.4.5.1: Workflow: Selecting a user from the main screen

6.4.5.2 Selecting a user from the list of nearby users

In order to initiate a new conversation or continue an existing one with a nearby user, the local users can simply navigate to the screen that displays the nearby users that was alluded to in subsection 6.4.3 and click the user they want to have a conversation with. This would lead to the presentation of a chat screen associated with the local user and the selected user similar to the one described in the preceding section. Figure 6.4.5.2 shows the workflow of selecting a user from the list of nearby users.



1. List of nearby users 2. Click on the user to chat with 3. Chat with the selected user

Figure 6.4.5.2: Workflow: Selecting a user from the list of nearby users

6.4.6 Sending a Message

Once a user is selected and the chatroom screen is presented, the local users can send messages to the selected user using the input controls on the bottom of the screen. Messages can be either text or images.

6.4.6.1 Sending a Text Message

To send a text message to the selected users the local users can simply fill in the text box on the bottom of the screen and subsequently click the send button adjacent to the textbox. Once the message is sent successfully the message would be displayed in the chat log portion of the local user's screen as well as the chat log of the selected user. Figure 6.4.6.1. Illustrates the workflow of sending a text message to the selected user.

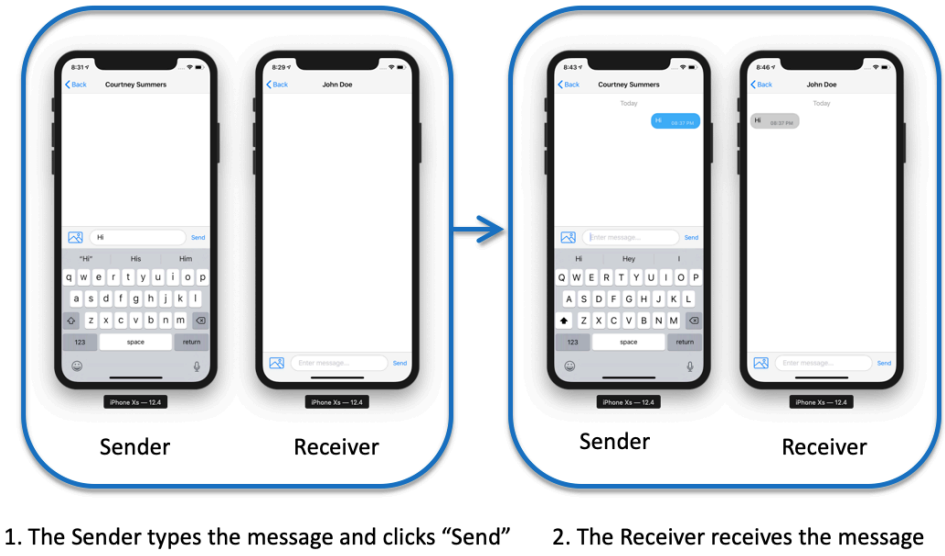


Figure 6.4.6.1: Workflow: Sending a text message to the selected user

6.4.6.2 Sending an Image

To send an image to the selected user the local user can click the button with the image icon on the left of the input text box. On clicking this button, the user would be asked to select the source from which the user wants to select the image. The user can select an image from the device’s photo library or click an image using the device’s camera. After choosing an image the user can simply click choose which would then initiate the process of sending the image to the selected user. Once the image is sent successfully the image would be displayed in the chat log portion of the local user’s screen as well as the chat log of the selected user. Figure 6.4.6.2. Illustrates the workflow of sending an image message to the selected user.

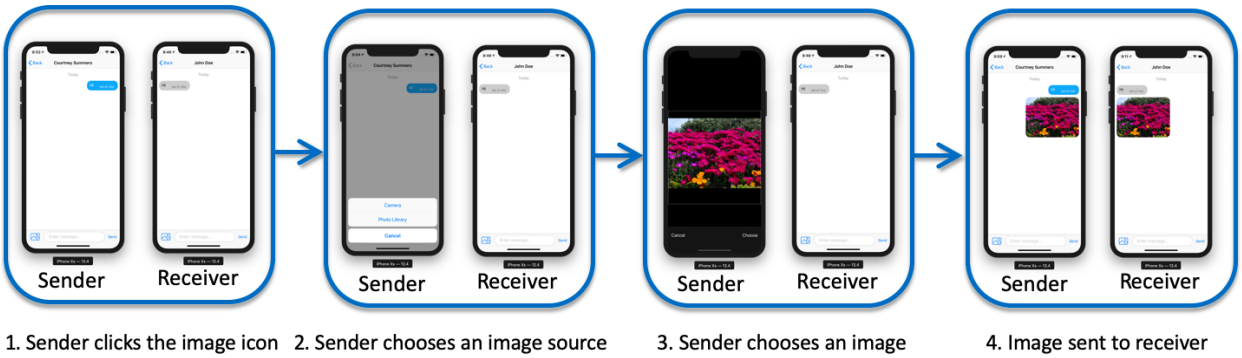


Figure 6.4.6.2: Workflow: Sending an image message to the selected user

6.4.7 Classifying and Masking Pornographic Images

If a sent image is identified as pornographic in nature, the image is masked at the recipient’s end (in the recipients chat log) using a blurred overlay that contains a message notifying the recipient that the image may contain pornographic content. This message also contains the confidence with

which the image was classified as being pornographic in nature by the deployed deep learning model. Masked images could be seen by their recipients (if they wish to) by clicking the button on the bottom of the masked overlay titled “See Photo” which would trigger the blurred overlay to disappear and the masked image to be displayed (unmasked). By the implementation of this mechanism the application ensures that any identified pornographic images sent to a user would only be viewable to that user at their own discretion after they have been notified of their possibly pornographic nature. Figure 6.4.7 diagrammatically represents a scenario where a user is sent a pornographic image by another user.

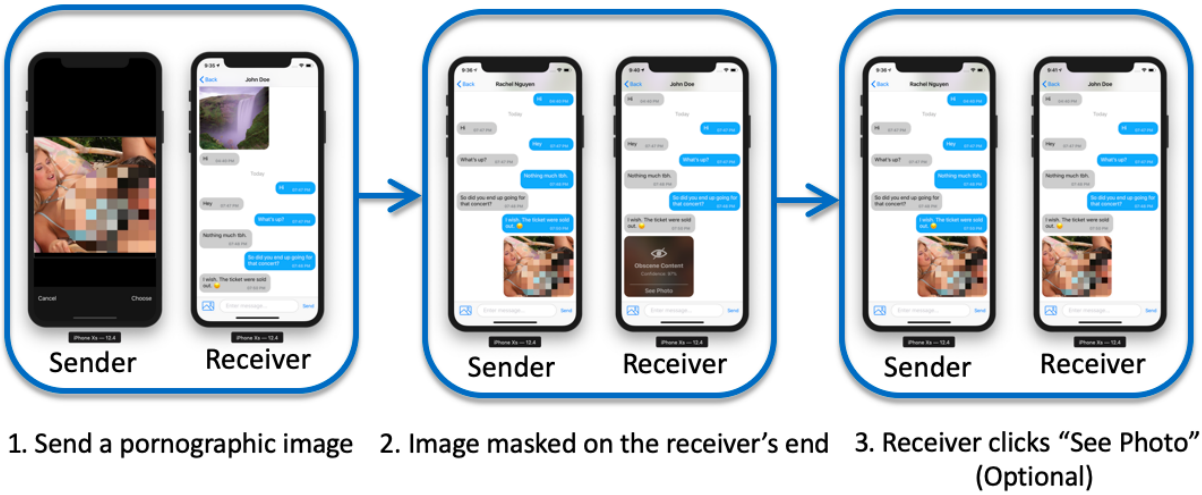


Figure 6.4.3.7: Scenario: Sending a pornographic image

6.4.8 Deleting Conversations

The application provides users with the capability to delete conversation that they have had with other application users. The application’s main screen (as shown in figure 6.4.3.1) list every existent conversation between the local user and other application users. In order to delete a conversation:

i. Navigate to the application’s main screen
ii. Locate the row representing the conversation to be deleted.
iii. Perform a right swipe screen gesture on the located row.
iv. Click the red delete button that appears on account of the swipe.

Figure 6.4.8 details the workflow to delete a conversation. If the local user deletes a conversation between itself and user X, the conversation would be deleted for the local user, but it would still be viewable to user X.

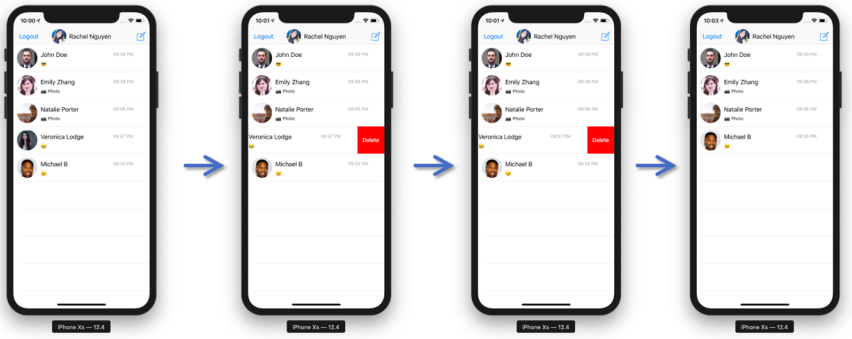


Figure 6.4.8: Workflow: Deleting a conversation

6.4.9 User Logout

Authenticated users can log out of their accounts on a given device. This can be done by navigating to the applications main page and then clicking the top left button in the navigation bar titled "Logout". Once the logout process is completed the applications presents the user with the login and sign up screen. Figure 6.4.9 describes the logout workflow.

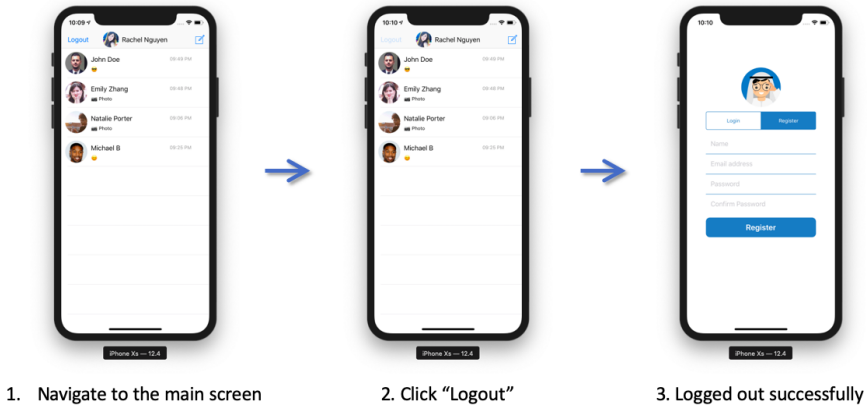


Figure 6.4.9: Workflow: Logout

6.4.10 Blocking User Accounts

The application's administrators possess the ability to block the accounts of users. This can be done through the firebase management console by going to the authentication section within the firebase project for the application, finding the user account to be blocked from the table of user accounts, clicking "Disable Account" from the options menu for that row and confirming the action. This would disable the users account precluding the user from logging in (as shown in the last screen of figure 6.4.2.2).

Chapter 7: iOS Future Work

7.1 Implement a Feedback Elicitation Subsystem

The Evaluator Application, as well as the Chat Application developed as part of this project, could be extended by implementing a mechanism to elicit user feedback pertaining to the results of the image classification. This could be used to keep track of misclassifications as well as to understand the behaviour and identify the idiosyncrasies of the model which could, in turn, help further improve the model.

7.2 Administrative Dashboard for Model Retraining and Version Control

An administrative dashboard could be developed to visualize and evaluate the performance of the models. Additionally, as an extension of the aforementioned feedback elicitation system, the dashboard could provide the functionality to view user feedback. Furthermore, the dashboard could also provide functionality for triggering model retraining based on this feedback. Model retraining could be triggered by the administrators when sufficient misclassified images start to accumulate since the model was last retrained. Retraining overtime would result in multiple versions of the model and ergo to further augment the capabilities of the system a model version control system could be implemented.

7.3 Augment Appropriate Security Mechanisms

The developed chat application stores data in an unencrypted format which is not the most appropriate mechanism as user privacy is of significant importance in social applications. End-to-end encryption could be implemented to make the application more secure. Additionally, images stored using the storage service use publicly accessible URLs for subsequent retrieval which poses several security concerns. One way to secure image retrievals would be by the usage of authentication tokens to ensure that only legitimate users that are authorised to access certain images are allowed to do so.

7.4 Implementation of more Reliable and Scalable Subsystems

To make the chat application more reliable and scalable the image storage and classification services can store images in a cluster-based database system that stores data redundantly as opposed to storing images on the filesystem of a single cloud instance. Additionally, load balancing could be used to adjust the number of cluster nodes based on the varying load. Serverless solutions could also be explored to store and classify images using functions (FaaS) for auto-scalability.

Chapter 8: Conclusion

In conclusion, the report attempted to explore deep learning for the classification of pornographic images while demonstrating its application in social systems (such as social media and chat applications). The project was divided into two phases. Phase 1 dealt with examining deep learning for adult image classification from a general standpoint while Phase 2 dealt with using the findings of Phase 1 as a basis to develop a mobile app to demonstrate the application of deep learning in social systems for the identification of pornographic content.

In the first phase, prominent deep learning models namely Inception v3, MobileNet v1, and MobileNet v2 were retrained using the NPDI pornographic dataset to identify pornographic images. A mobile application was developed to deploy these retrained models using different deployment paradigms (on-device and cloud-hosted). Subsequently, the various models and deployment paradigms were evaluated using different evaluation metrics. The metrics used to evaluate the performance of the retrained models were the training and validation accuracy, the cross-entropy loss, the average confidence, and the F1-score. Whereas, the metrics used to compare deployment paradigms were the size of the model and the inference time.

In the second phase of this project, a real-time iOS Chat Application that is capable of identifying and masking pornographic images was developed as a way to proffer a potential solution to the unsolicited pornographic image dissemination problem that was identified to be prevalent in modern social systems. Relevant literature coupled with the findings and insights obtained from Phase 1 of the project were drawn upon to make vital decisions pertaining to the architecture and design of the application. The cloud-hosted Inception v3 was deemed the most ideal among the explored model and deployment paradigm combinations for the purposes of developing the application. The retrained Inception v3 model provided an accuracy of 86.3% and an F1-Score of 0.836. While the Inception v3 cloud-hosted model and deployment paradigm combination provided an average inference time of 1743 ms.

In the future, the capacities of the developed system can be augmented by the development of a feedback elicitation subsystems and an administrative dashboard for model retraining and version control. Additionally, colossal userbases coupled expectations of high performance and uninterrupted availability make scalability and reliability paramount non-functional requirements in social applications. Furthermore, the growing concerns about user privacy in such applications make security an important concern as well. As a result, mechanisms that augment greater scalability, reliability, and security can be implemented to make the application more practical for the deployment in a production environment.

References

[1] Dong-Hun, L. (2010). Korean consumer & society: Growing popularity of social media and business strategy. SERI Quarterly, 3(4), 112-117,9. Retrieved from http://search.proquest.com.ezp.lib.unimelb.edu.au/docview/758867913?accountid=12372

[2] Goldman, Phillip Y. "Practical techniques for reducing unsolicited electronic messages by identifying sender's addresses." U.S. Patent No. 7,516,182. 7 Apr. 2009.

[3] BusinessToday.In, "WhatsApp users share 55 billion texts, 4.5 billion photos, 1 billion videos daily," *Businesstoday.in*, 27-Jul-2017. [Online]. Available: https://www.businesstoday.in/technology/news/whatsapp-users-share-texts--photos-videos-daily/story/257230.html. [Accessed: 05-Nov-2019].

[4] salman.aslam.mughal, "• Snapchat by the Numbers (2019): Stats, Demographics & Fun Facts," *Omnicoreagency.com*, 06-Jan-2019. [Online]. Available: https://www.omnicoreagency.com/snapchat-statistics/. [Accessed: 19-Feb-2019].

[5] Crawford, K, Gillespie, T (2014) What is a flag for? Social media reporting tools and the vocabulary of a complaint. New Media & Society. Epub ahead of print 15 July. DOI: 10.1177/1461444814543163.

[6] West, S. M. (2018). Censored, suspended, shadowbanned: User interpretations of content moderation on social media platforms. New Media & Society. Advanced online publication. doi:10.1177/1461444818773059

[7] Jones, M.J., and Rehg, J.M. Statistical color models with application to skin detection. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., vol. 1. IEEE (1999)

[8] Avila, S., Thome, N., Cord, M., Valle, E., and Araujo, A., Pooling in image representation: The visual codeword point of view. Computer Vision and Image Understanding, 117(5), pp. 453 – 465 (2013)

[9] M. Moustafa, "Applying deep learning to classify pornographic images and videos," ArXiv151108899 Cs, Nov. 2015

[10] W. A. Arentz and B. Olstad, "Classifying offensive sites based on image content," Comput. Vis. Image Underst., vol. 94, no. 1–3, pp. 295–310, Apr. 2004.

[11] M. M. Fleck, D. A. Forsyth, and C. Bregler, "Finding naked people," in Computer Vision — ECCV '96, B. Buxton and R. Cipolla, Eds. Springer Berlin Heidelberg, 1996, pp. 593–602.

[12] Q.-F. Zheng, W. Zeng, W.-Q. Wang, and W. Gao, "Shape-based adult image detection," Int. J. Image Graph., vol. 6, no. 1, pp. 115–124, Jan. 2006.

[13] T. Deselaers, L. Pimenidis, H. Ney, Bag-of-visual-words models for adult image classification and filtering, in: Proceedings of the International Conference on Pattern Recognition (ICPR), 2008, pp. 1–4.

[14] S. Indolia, A. Goswami, S. Mishra and P. Asopa, "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach", Procedia Computer Science, vol. 132, pp. 679-688, 2018. Available: 10.1016/j.procs.2018.05.069.

[15] Z. Li, X. Zhu, L. Wang, and P. Guo, "Image Classification Using Convolutional Neural Networks and Kernel Extreme Learning Machines," 2018 25th IEEE International conference of Image Processing (ICIP), 2018.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2017.

[17] Understanding of Convolutional Neural Network (CNN) — Deep Learning", Medium, 2019. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Accessed: 21- Oct- 2019].

[18] Agostinelli, F., Hoffman, M., Sadowski, P., Baldi, P.L., 2014. Learning activation functions to improve deep neural networks. ArXiv preprint arXiv:1412.6830.

[19] M. Ranzato, Y.-L. Boureau, and Y. LeCun. Sparse Feature Learning for Deep Belief Networks. In NIPS, 2007

[20] V. Kumar, “21 Artificial Intelligence Tools To Make Your Project More Effective,” RankRed, 20-Feb-2019. [Online]. Available: https://www.rankred.com/artificial-intelligence-tools/. [Accessed: 21-Oct-2019].
[21] J. Cao, Z. Su, L. Yu, D. Chang, X. Li and Z. Ma, "Softmax Cross Entropy Loss with Unbiased Decision Boundary for Image Classification,
" *2018 Chinese Automation Congress (CAC)*, Xi'an, China, 2018, pp. 2028-2032.
doi: 10.1109/CAC.2018.8623242
[22] Taddy, Matt (2019). "Stochastic Gradient Descent". Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions. New York: McGraw-Hill. pp. 303–307. ISBN 978-1-260-45277-8.
[23] I. Goodfellow, Y. Bengio and A. Courville, Deep learning. Cambridge (EE. UU.): MIT Press, 2016.
[24] P. Marcelino, “Transfer learning from pre-trained models,” Medium, 23-Oct-2018. [Online]. Available: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751.
[25] Matthijs Hollemans, “MobileNet version 2,” Machinethink.net, 2018. [Online]. Available: https://machinethink.net/blog/mobilenet-v2/. [Accessed: 01-Nov-2019].
[26] “MobileNetV2: The Next Generation of On-Device Computer Vision Networks,” Google AI Blog, 03-Apr-2018. [Online]. Available: https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html. [Accessed: 01-Nov-2019].
[27] C. SZEGEDY, V. VANHOUCKE, S. IOFFE, J. SHLENS, AND Z. WOJNA, Rethinking the inception architecture for computer vision, arXiv preprint arXiv:1512.00567, (2015).
[28] “Advanced Guide to Inception v3 on Cloud TPU | Cloud TPU | Google Cloud,” Google Cloud, 28-Jan-2019. [Online]. Available: https://cloud.google.com/tpu/docs/inception-v3-advanced.
[29] “AutoML for large scale image classification and object detection,” *Google AI Blog*, 02-Nov-2017. [Online]. Available: https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html. [Accessed: 05-Nov-2019].
[30] Sik-Ho Tsang, “Review: NASNet—Neural Architecture Search Network (Image Classification),” *Medium*, 18-May-2019. [Online]. Available: https://medium.com/@sh.tsang/review-nasnet-neural-architecture-search-network-image-classification-23139ea0425d. [Accessed: 05-Nov-2019].
[31] “TensorFlow Core,” TensorFlow, 2019. [Online]. Available: https://www.tensorflow.org/overview. [Accessed: 01-Nov-2019].
[32] S, C. (2019). *Intro to Firebase 2.0: Mobile-Backend-as-a-Service (MBaaS) from Google*. [online] Meetup. Available at: https://www.meetup.com/en-AU/acloud/events/rtntpyvjbsb/ [Accessed 5 Nov. 2019]
[33] “What Distinguishes Erotica From Pornography?,” *Psychology Today*, 2011. [Online]. Available: https://www.psychologytoday.com/au/blog/evolution-the-self/201104/what-distinguishes-erotica-pornography. [Accessed: 05-Nov-2019].
[34] “Database | JavaScript SDK | Firebase,” *Firebase*, 2019. [Online]. Available: https://firebase.google.com/docs/reference/js/firebase.database.Database. [Accessed: 05-Nov-2019].
[35] D. East, “Client-side fan-out for data consistency,” *The Firebase Blog*, 07-Oct-2015.
[36] projects.locations.functions, “REST Resource: projects.locations.functions | Cloud Functions | Google Cloud,” *Google Cloud*, 26-Jun-2019. [Online]. Available: https://cloud.google.com/functions/docs/reference/rest/v1/projects.locations.functions. [Accessed: 05-Nov-2019].

Appendices

Appendix A:
Source Code Link: https://github.com/knightmatish/DeepLearningProject.git (Code Only)

Appendix B:
Google Drive Link: https://drive.google.com/drive/folders/1lMl_c7DqjZyRS5T6-WsSKjCi4EdQphL-?usp=sharing
(Full)

Appendix C:
Demonstration of the Evaluator Application: https://www.youtube.com/watch?v=7e12PeKqJc4

Appendix D:
Demonstration of the Chat Application: https://www.youtube.com/watch?v=eJMeyl9erA4