iLieDown: Improved Display Orientation for Handheld Devices Using Convolutional Neural Networks

Riley Tallman
Arizona State University
rtallman@asu.edu

Abstract

91% of smartphone and tablet users experience a problem with their device screen being oriented the wrong way during use [11]. In [11], the authors proposed iRotate, which used computer vision to solve the orientation problem. We propose iLieDown, an improved method of automatically rotating smartphones, tablets, and other device displays. This paper introduces a new algorithm to correctly orient the display relative to the user's face using a convolutional neural network (CNN). The CNN model is trained to predict the rotation of faces in various environments through data augmentation, uses a confidence threshold, and analyzes multiple images to be accurate and robust. iLieDown is battery and CPU efficient, causes no noticeable lag to the user during use, and is 6*x* more accurate than iRotate.

1 Introduction

The current basic method to orient handheld displays on devices like smartphones and tablets is to use gravity. Unfortunately, this method fails when a user lies down to one side, or when the device is flat as Figure 1(a) illustrates. In a survey of 513 smartphone and tablet users from 2012, 91% reported that they have experienced incorrect autorotation with their devices, and 42% answered that they experience it several times a week or more [11]. To combat this issue, devices commonly have a native software feature to lock their orientation. For example, the original Apple iPad (released in 2010) has a hardware switch to lock and unlock the native orientation. This solution requires user input, and 58% of users forget to unlock the orientation [11]. These statistics show a need for a solution.

We propose iLieDown which correctly orients a display requiring no user input. This paper introduces a new robust algorithm in section 2 which analyzes multiple images from the front-facing camera using a VGG-style convolutional neural network (CNN) [13].

iLieDown yields correct orientation because it rotates the display relative to the user's face and how the device is being used rather than simply relative to gravity. Experiments in section 3 show that iLieDown is battery efficient, robust, and does not introduce noticeable lag when rotating.

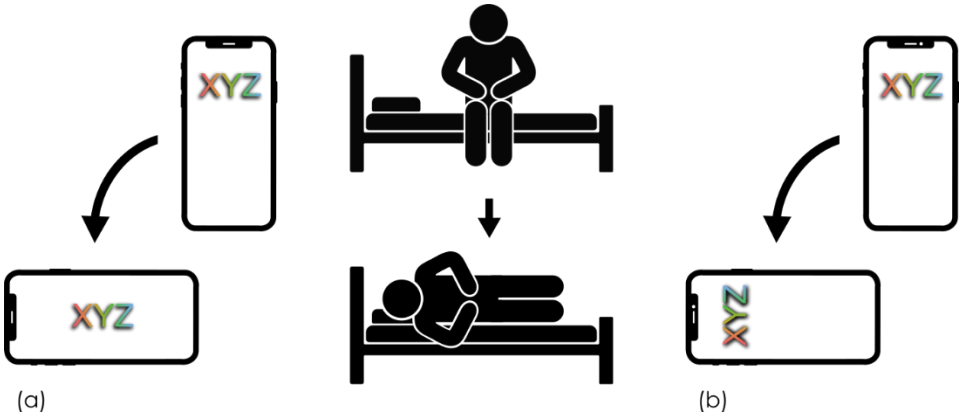

Figure 1: **Correct vs Incorrect Rotation.** The user lies down and the screen rotates incorrectly in (a), but correctly in (b).

2 AUTOMATIC FACE ROTATION

We present iLieDown which uses gravity to detect rotation changes, then uses a CNN to provide correct orientation. It is possible to constantly analyze images using a CNN without gravity-based input, but this is inefficient as Table 4 shows. An efficient approach is to only analyze images when a change in rotation of the device is detected by using the accelerometer. Unfortunately, this method cannot orient a display when the device is flat because gravity cannot be used to detect rotation changes.

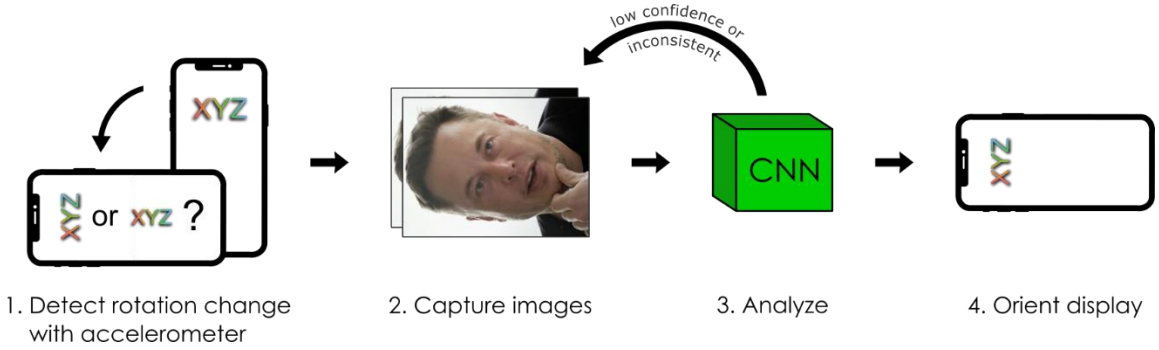

Figure 2: **iLieDown Overview.** Rotation is detected relative to gravity, then 2 images are analyzed by the CNN. If the analyses have low confidence or inconsistency, more images are captured and analyzed, otherwise, the display is oriented correctly according to the user's face.

Once a rotation change is detected, it is immediately known that there are only two possible orientations: the previous orientation or one relative to gravity (*portrait* or *landscapeLeft* in Figure 2, step 1). This is because there is always only one orientation relative to gravity, and there is the possibility that the user has lied down in which the display should remain at the previous orientation. The next step is to capture images from the front-facing camera and analyze them with a CNN. At this point, more images are captured and analyzed if there is low confidence or inconsistency in the analysis. Finally, iLieDown either confirms that the gravity orientation is correct or will override gravity orientation and keep the previous orientation. With this method, the user can lie down and maintain correct orientation on their device without explicit input.

2.1 CUSTOM CNN

We've created and trained a custom convolutional neural network to identify the face of the user and thereby determine correct orientation. Orienting a display is framed as a 4-class classification problem to classify a single input image as one of four orientations: *portrait*, *upsidedown*, *landscapeLeft*, and *landscapeRight*.

Architecture Design. The CNN architecture is most important for engineering efficient image analysis models. The size and efficiency of CNNs are characterized by the number of parameters they have, which depends on the architecture. The number of parameters directly influences the number of calculations that are executed during image analysis, and more calculations take more time. A CNN that is too big (has too many parameters) will not be able to analyze images quickly. On the other hand, a CNN that is too small will not learn very well and will have many misclassifications [14].

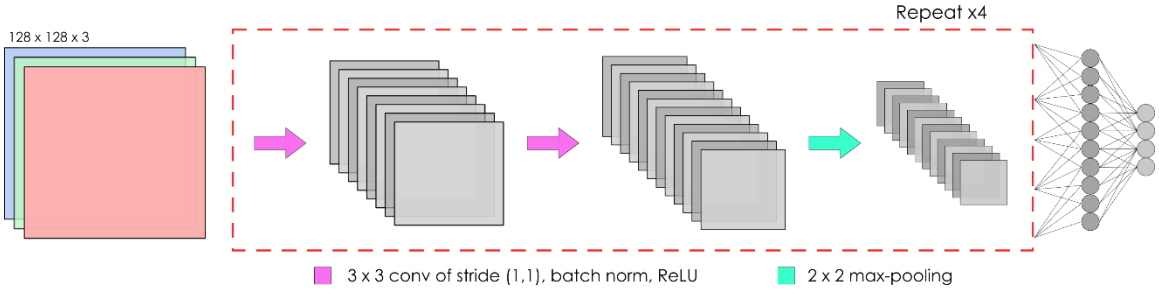

Figure 3: **Network Architecture.** Images are scaled once to 128x128 pixels then fed through a series of 3x3 convolutions with a stride of 1 in both x and y directions, batch normalization, and ReLU activation. After max-pooling, those layers are repeated 4 times. The last two layers are fully connected.

Our network needs to be small enough to quickly analyze images so that the user does not notice lag before rotation. Figure 3 shows our implementation of a VGG-B style [13] network which is composed of four iterations of two convolutional layers followed by a

single max-pooling layer. This architecture was selected because of its small size and simplicity.

Training. We modified the Helen dataset [15] to train our network because the majority of images had faces that were looking at the camera lens, which is like how a user would be looking at a smartphone or tablet's display while using it. To create our training data, every image was flipped horizontally, then rotated three times. Finally, the images were scaled down to 128x128 pixels. Additional data augmentation was used during training such as minor random rotations and brightness alterations. The images were also linearly blurred to simulate the user shaking their device while rotating. These techniques create a robust training set which allows our model to generalize well and increase accuracy.

The model was trained using categorical cross-entropy, so it outputs four numbers that sum to 1, each representing the confidence of the image belonging to its class (examples in Table 2). During training, a dropout layer [16] was used after each max-pooling layer to prevent overfitting. Our final network achieved 0.997 validation accuracy with 23,090 parameters.

2.2 ALGORITHM DETAILS

Algorithm 1 shows the core logic of how iLieDown makes decisions. It is called when the device changes rotation relative to gravity.

Algorithm 1 Classifier with Confidence Threshold and Multiple Image Analysis

```
1:  imageQueue = analyzeImages(x − 1)
2:  do:
3:      imageQueue += analyzeImages(1)
4:      override = true
5:      for image in imageQueue:
6:          if image.confidence < c or isInconsistent(image.prediction):
7:              override = false
8:      if override:
9:          return 'OVERRIDE'
10:     imageQueue.pop
11: while (not analyzed n images)
12: return 'Low Confidence or Inconsistent'
```

Confidence. Misclassification may occur if an image is terribly blurry, dark, or bright, or if the user's face is obstructed. To combat this, we define a confidence threshold $\boldsymbol{c}$ on line 6. In order for the CNN to override gravity-based orientation, it needs to reach a confidence greater than some threshold $\boldsymbol{c}$ on two consecutive images that it analyzes. If the

first two images fail to meet the confidence threshold, then the program will continue to analyze images from the front camera up to $\boldsymbol{n}$ images.

Consistency. The *isInconsistent* function ensures that the current image has predicted an orientation consistent with the other predictions from images in the *imageQueue*: if the orientation differs, then it is not consistent. The function also verifies that the prediction is one of the two possible orientations out of four because only two orientations are possible when rotating: relative to gravity and previous orientation. This is depicted in step 1 of Figure 2 when iLieDown knows that either a *landscapeLeft* or *portrait* orientation is possible.

ImageQueue. iLieDown analyzes a set of images (the *imageQueue*) to ensure correctness. This technique is analogous to using multiple keys to authorize a military strike in movies: all keys need to work in order to execute the operation. The parameter $\boldsymbol{x}$ specifies how many images, or keys, to use in order to override gravity. The loop on line 5 ensures that all images in the *imageQueue* are above the confidence threshold and are consistent. Setting $\boldsymbol{x}$ greater than 1 helps prevent misclassifications.

Multiple Images. To be robust, the algorithm may analyze up to $\boldsymbol{n}$ images than are in the initial *imageQueue*. When the CNN is not confident or if the predictions are inconsistent, an image in the *imageQueue* is discarded and one new image is analyzed. This creates a sliding window that shifts over one image in the camera feed up to $\boldsymbol{n}$ times. A total of $\boldsymbol{n}$ images may be analyzed, but "OVERRIDE" is returned (line 9) as soon as iLieDown is confident with its prediction. Thus, as few as $\boldsymbol{x}$ images will be analyzed.

Model	Class Confidence				Result
	portrait	*upside down*	*landscape Left*	*landscape Right*	
Gravity				1	OVERRIDE: ***portrait***
CNN (**1**)	**0.93**	0.05	0.01	0.01	
CNN (**2**)	**0.93**	0.06	0.01	0.00	
Gravity				**1**	Low Confidence: ***landscapeRight***
CNN (**1**)	0.54	0.23	0.11	0.12	
CNN (**2**)	0.56	0.20	0.11	0.13	
CNN (**3**)	0.51	0.26	0.11	0.12	

Table 2: **Sample Outputs and Results from Algorithm 1.** Parameters: $x = 2$, $c = 0.75$, $n = 3$. CNN(1) is the first analyzed image, and CNN (2) is the second.

2.3 EXAMPLE OUTPUT

iLieDown either returns "Low Confidence or Inconsistent" and the display is oriented based on gravity, or returns "OVERRIDE" and gravity is overridden. In Table 2, parameter $\boldsymbol{x}$ equals 2 which means two successive images must meet threshold and consistency requirements. In the first sample output, the result is an override because both images are above a confidence of 0.75 and are consistent. In the second example however, the first two analyzed images are below the confidence threshold. When this happens, a third image is analyzed and the first one is dropped. Unfortunately, the third image has low confidence as well, so the CNN predictions are discarded, and iLieDown falls back on gravity orientation. No more images are analyzed because n is set to 3, which is the maximum number of images to analyze before giving up and using gravity.

3 EXPERIMENTS

We tested iLieDown in real-world environments to gauge its effectiveness. Using 20 participants (14 male) we organized two tests and asked survey questions.

Speed test. In order to determine if there was any noticeable lag introduced with Automatic Face Rotation, users tested the speed of rotations on two apps: one with native rotation in iPhone Operating System (iOS) 12.4 (presumably gravity) and one running iLieDown. Users were asked to identify if they were the same speed or if one was faster than the other on an iPhone 6 and an iPhone XS.

	Native is faster	iLieDown is faster	Same speed
iPhone 6	7	0	13
iPhone XS	3	2	15

Table 4. **Noticeable Lag Responses.** Users compared the rotation speed of native rotation vs iLieDown. Parameters $x, n,$ and c were set to 2, 8, and 0.75 respectively (see Algorithm 1).

The majority of users were unable to determine which app was faster, especially on the iPhone XS with its faster processor. From the results in Table 4, we conclude that iLieDown introduces no noticeable lag to the user on an iPhone XS, and that most users will not be able to notice lag on an iPhone 6.

Of course, iLieDown does in fact take more time because of the additional image processing. On average, it takes 32 milliseconds (ms) to analyze a single image on an iPhone 6, and just 9ms on an iPhone XS (Table 5). We asked users if they would enable or disable iLieDown knowing that there may be some minor lag during each rotation and all 20 happily users responded that they would enable the feature.

Real-World Performance. Users were instructed to hold the device in portrait and then lay down where it was recorded if iLieDown oriented the display correctly. Then, still laying to one side, the user successively rotated the device 4 times and read a portion of a pre-loaded webpage at each rotation. This was then repeated, laying down to the other side. Data was collected for each rotation. To best simulate real-world use, users were unaware of the camera being used and they were given a webpage with text to read and scroll as necessary. Furthermore, tests were performed in random locations on couches and beds to provide various backgrounds and lighting conditions for the CNN. Overall, iLieDown correctly oriented the display in 61% of trials. This is near a 6*x* improvement over the previous solution from [11] who reported 10.7% accuracy with similar tests.

Privacy Survey. We asked users if they would use Automatic Face Rotation (as if it were programmed into the operating system) knowing that the front camera takes pictures every time the device is rotated. 18 users said they would use it, and 2 said it depended on the device manufacturer's reputation since some companies have a history of negligence and/or intentional misuse of users' data.

We then asked if users would allow random anonymous image data to be collected to improve iLieDown. 10 users said yes, 3 said it depended on the manufacturer's reputation, and 6 said they would not allow it. One responded that they would allow it only if they could choose what pictures were sent. For example, a notification to review the images would allow them to deny or allow them to be sent.

CNN Speed. We timed the exact speed of our custom CNN (23,090 parameters) by recording the milliseconds it took to analyze 2, 5, and 8 images. Table 3 shows the average of those values over 20 trials. Note that the iPhone 6 had a setting that reduced peak performance on the processor. Two tests were conducted with this setting on and off.

CPU and Memory Usage. We collected data about the CPU and memory usage using Xcode 10.3. In Table 3, Max CPU shows what percentage of the processing power iLieDown uses when it runs during a rotation with n set to 2 and 8 images.

	Average time to analyze images (milliseconds)			
	2	5	8	1*
iPhone 6~	65	163	260	33 ± 2
iPhone 6	66	158	263	33 ± 2
iPhone XS	20	47	77	9 ± 2

Table 5. **Average time taken to analyze images.** The time in milliseconds to analyze 2, 5, and 8 images averaged over 20 trials. **Average time per image plus or minus the standard deviation across all trials.* ~*Battery health at 87% maximum capacity and "performance management applied"*

	Default Camera*	Analysis at 30 FPS~	Max CPU+	
			2 images	8 images
iPhone 6	1.5%	20%	4.5%	7.5%
iPhone XS	0.5%	7%	1.5%	2.2%

Table 3. **CPU Usage of iLieDown.** **Capturing images from the front camera at 30 frames per second (FPS) with default settings in iOS 12.4. ~Average over 2 minutes during constant analysis of 30 FPS. +The maximum amount of CPU used during a single rotation.*

4 RELATED WORK

Our work is most related to the work of [11] who presented iRotate: an automatic screen rotation method by augmenting the gravity-based solution. They used the same technique of running a facial detection algorithm when a rotation change is detected and orienting the screen based on the user's face. Instead of creating a custom CNN, they used a pre-built facial feature detection API from iOS 5. The API was not well suited for the application, and the authors deemed it infeasible to use after reporting 10.7% success rate with 20 participants. This paper builds on their solution and proposes a custom CNN and introduces a novel algorithm for analyzing multiple images. Nonetheless, the authors presented great statistics about the when and how frequently incorrect rotation occurs.

Another camera-based approach attempted to rotate a display by tracking the head tilt of user's faces [6]. Unfortunately, this suffered from a narrow range of degree tilt (about 47 degrees away from *portrait* in total) [7].

Gravity-based solutions are popular, and various types of sensors have been used including mercury switches [1] and one or more accelerometers [2, 3, 8]. These give incorrect rotations when a user lies down or when the device is flat, but offer a manual toggle to correct it. Our solution fixes the problem when a user lies down, and requires no user input.

Further methods use the way a user holds their phone to distinguish orientations. One solution explores the position of thumbs on the edges of the front screen [9], and another senses an entire grasp from the placement of fingers around the edges and the palm on the back [10]. iRotateGrasp was one such method that used 44 capacitive sensors to Figure out the grasp of the user. The implementation was successful, and when tested against 6 users it achieved 80.9% average validation accuracy and 90.4% average accuracy when trained specifically to one user. This solution will work in complete darkness, but it requires additional sensors located on the outside of devices.

5 DISCUSSION AND FUTURE WORK

We presented a method that correctly orients displays on handheld devices without user input. The proposed framework, iLieDown, uses a convolutional neural network and a novel algorithm to analyze images from the front-facing camera. It introduces no noticeable lag compared to the native rotation in iOS 12.4 and remains battery efficient given limited resources. In real-world tests, iLieDown has decent accuracy in various environments and lighting conditions.

Automatic Face Rotation can likely be improved with additional pre-processing techniques on captured images such as edge detection and noise reduction. This would allow the CNN to be more robust towards various lighting conditions.

In our custom CNN we classified all four orientations, however, the *upsidedown* orientation can most likely be omitted in the future. We asked 20 users how often they use their devices *upsidedown*, and just 1 user responded about once a week. By omitting *upsidedown*, there will be fewer classes to predict and the CNN can be smaller and therefore more efficient.

An alternative solution to using a CNN could be to use an infrared (IR) camera. IR cameras can capture information in extreme low-light environments and can capture depth of objects. Newer devices already use this technology to analyze faces for security [4,5].

The second most common source of incorrect rotation is when a device is flat [11]. In this situation, gravity cannot be used to detect rotation changes. An algorithm to periodically analyze images from the front camera might be able to solve this issue.

REFERENCES

1. Schmidt, Albrecht, Michael Beigl, and Hans-W. Gellersen. "There is more to context than location." *Computers & Graphics* 23, no. 6 (1999): 893-901.
2. Bartlett, Joel F. "Rock'n'Scroll is here to stay [user interface]." *IEEE Computer Graphics and Applications* 20, no. 3 (2000): 40-45.
3. Hinckley, Ken, Jeff Pierce, Mike Sinclair, and Eric Horvitz. "Sensing techniques for mobile interaction." In *UIST*, vol. 2000, pp. 91-100. 2000.
4. Shimada, Hiroyuki, Yasuyuki Nakamura, and Yuichi Kageyama. "Infrared face authenticating apparatus, and portable terminal and security apparatus including the same." U.S. Patent Application 11/674,622, filed August 16, 2007.
5. Sung, Younghun, and Euihyeon Hwang. "Apparatus and method for human distinction using infrared light." U.S. Patent 7,536,037, issued May 19, 2009.
6. Blaskó, Gábor, William Beaver, Maryam Kamvar, and Steven Feiner. "Workplane-orientation sensing techniques for tablet PCs." In *Proceedings of the 17th Annual ACM symposium on User Interface Software and Technology*. 2004.
7. Bradski, Gary R. "Computer vision face tracking for use in a perceptual user interface." (1998).
8. Forstall, Scott, and Chris Blumenberg. "Portrait-landscape rotation heuristics for a portable multifunction device." U.S. Patent 7,978,176, issued July 12, 2011.

9. Ording, Bas, Marcel Van Os, and Imran Chaudhri. "Screen rotation gestures on a portable multifunction device." U.S. Patent 7,978,182, issued July 12, 2011.
10. Cheng, Lung-Pan, Meng Han Lee, Che-Yang Wu, Fang-I. Hsiao, Yen-Ting Liu, Hsiang-Sheng Liang, Yi-Ching Chiu, Ming-Sui Lee, and Mike Y. Chen. "iRotateGrasp: automatic screen rotation based on grasp of mobile devices." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3051-3054. ACM, 2013.
11. Cheng, Lung-Pan, Fang-I. Hsiao, Yen-Ting Liu, and Mike Y. Chen. "iRotate: automatic screen rotation based on face orientation." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2203-2210. ACM, 2012.
12. Kazemi, Vahid, and Josephine Sullivan. "One millisecond face alignment with an ensemble of regression trees." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1867-1874. 2014.
13. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
14. Iandola, Forrest, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. "Densenet: Implementing efficient convnet descriptor pyramids." *arXiv preprint arXiv:1404.1869* (2014).
15. Le, Vuong, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S. Huang. "Interactive facial feature localization." In *European conference on computer vision*, pp. 679-692. Springer, Berlin, Heidelberg, 2012.
16. Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15, no. 1 (2014): 1929-1958.