A Smart Way to Design Future Smartphones

Tz-Ruei Liu
tliu292@wisc.edu

Kyle Wang
kyle.wang@wisc.edu

Zhilin Wang
zwang2244@wisc.edu

Abstract

Nowadays, smartphones have become an essential part of people's lives. To satisfy customers' demands and earn more profit, manufacturers release new versions of smartphones frequently. This project focuses on smartphone layouts and generates creative smartphone designs using deep learning techniques. First, we collect more than 6500 online smartphone images, as well as classical mobile phone images. Several preprocessing methods are applied to manipulate the input training images, including edge detection, manually masking screens, and colorizing different features of a smartphone to possibly improve the results. After exploring multiple deep learning techniques, Deep Convolutional Generative Adversarial Networks (DCGAN) is chosen as the model for this study and transfer learning as an auxiliary technique. We first generate images by DCGAN without any preprocessing over the training set. However, the generated smartphones are not satisfactory and they are mostly covered with colorful wallpapers, and no obvious information about the position of phone parts such as cameras or buttons are revealed. Our second trial uses DCGAN with white-out screens and edge detection. Although we can find the positions of specific parts such as the speaker, the result is unsatisfying since no creative designs are generated. The later trials apply transfer learning but the results are still under expectation. Finally, we apply DCGAN to colorized smartphones; the result is better with respect to the positions of cameras, speakers, buttons and etc; however, we consider this result as not outstanding.

1. Introduction

For this recent decade, smartphone technologies have become so embedded in people's lives that we would have trouble doing virtually anything for one day without them. Besides core phone functions such as sending messages and making phone calls, smartphones are of multimedia functionality including playing music, taking photos, keeping track of schedules and appointments, gaming, and anything that you can think of. Therefore, the smartphone industry has been one of the most innovative and rapidly developing fields within Information and communications technology (ICT). Technical change and new product proliferation have made this industry extremely dynamic, even if market shares are highly concentrated in the hands of very few companies [1]. To make profits and improve their competitiveness in the market, these manufacturers keep renewing their design of mobile phones and launch new versions of smartphones frequently. The appearance of smartphones has been dramatically changed throughout the past twenty years. Before the 2010s, most smartphones have explicit number buttons for people to type. However, with the introduction of the iPhone to the mobile phone's market, there are seldom any number buttons on the phones; what is remaining is a large screen, a notch, and a headphone jack. A digital mode is preferred than a physical mode.

Figure 1. Smart phones [2]

In addition, many scientists and industry experts are predicting what mobile phones will look like in future years and decades. Some scientists predict that holographic displays and flexible frames would appear for future phones [3], and others believe that phones would have bigger screens. Similarly, we are interested in exploring the design choice of future phones using the deep learning tools, given the mobile phones' design and appearance in the past decade. In this project, we are going to design a smartphone appearance by deep learning techniques to see if these models can generate some surprisingly creative designs and break the "traditional" design. We could compare our predicted models and future smartphones to see whether

Figure 2. Chair Design

our prediction is accurate when it's released.

2. Related Work

There have been some related works done previously that use deep learning methods to design transitional items, such as chairs. The chair project [4] by Philipp Schmitt is a similar work focusing on designing a chair by deep learning models. A DCGAN model [5] using 600 images of chairs is trained and is able to produce hundreds of designs. Then, they turned the blurry generated images of chairs to sketches and by combining with physical restrictions to make those designs in physical forms. Those physical chairs are presented to designers for chairs, so that hopefully they can be inspired. Figure 2 is an example of chairs designed by the DCGAN model.

There's another project done by a group from PES University in India that they use GAN to generate designs for cars [6]. They collocted 100,000 images of different cars and, similar to this project, they spent quite a long time to do the preprocessing. They use another Machine Learning model trained with those famous large image datasets to remove undesirable objects in the original images. Then they spent additional large amount of time, which is also similar to this project, to focus on manually inspecting those remaining abnormal artifacts in the dataset. As they aimed to help designers to produce car models based on their sketches, the goal for their edge detection is different from this project's. They tried with Canny Edge Detection first, which is also used in this project, and it failed to capture nuance in human's sketches. As a result, they eventually came up with their own BiSECT Sketching Filter to include the details. Later on, they trained a Conditional GAN model with 200 epochs and 200 paired images to "teach" the generator how to produce prototypes from sketches. Figure 3 shows an example of their results that by inputting the sketch of a car, a colorized car design can be generated.

We are motivated by a image preprocessing way used in the paper "House-GAN: Relational Generative Adversarial

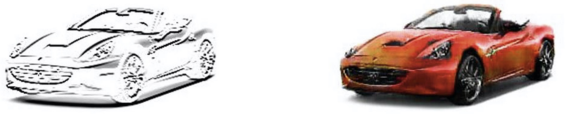

Figure 3. Example of Car Project's result

Networks for Graph-constrained House Layout Generation" by Nauata et al [7]. They build a graph-constrained generative adversarial network to generate a new house layout. The most interesting part is that they preprocess a set of house layouts by painting each room with a color based on its type to help the GAN to recognize the room type. An example is shown in the Figure 4. In this way, the GAN can actually learn about the more specific features of layouts of houses, compared with directly being trained with raw images of house layouts. Not only the way rooms are arranged can be well presented in this way, but also the size of each room can be shown. We also apply this technique in our project that we paint the screen with green, cameras with blue, buttons with white and the speaker with yellow to help DCGAN recognize the positions of different phone parts as well as sizes of each part.

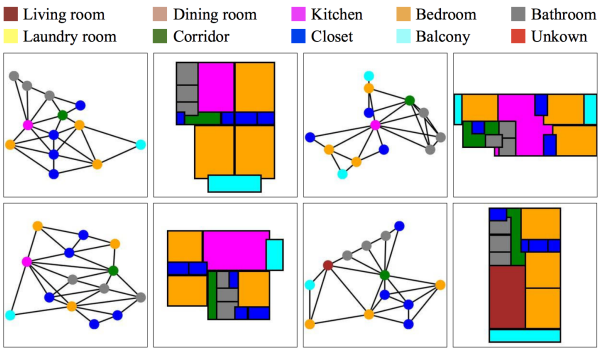

Figure 4. Sample bubble diagrams and house layouts.

3. Proposed Method

As what has been done for the project about designing chairs, DCGAN is chosen to be the GAN model to train and test in this study. DCGAN is shorted for deep convolutional generative adversarial networks. A general GAN can capture the training data's distribution and generate new data from this distribution. Additionally, a DCGAN is an extension to general GAN, which uses convolutional and convolutional-transpose layers in the discriminator and generator [5]. Figure 5 is a DCGAN generator.

PGAN is also taken into consideration [8]. However, the required input image is in size 512 * 512, which will require

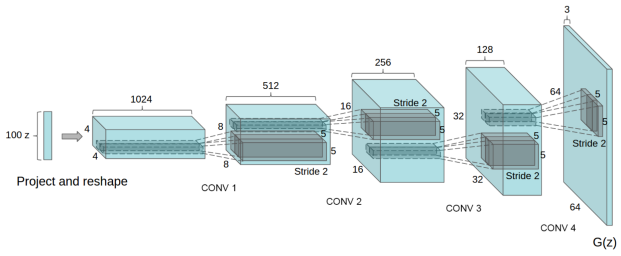

Figure 5. DCGAN generator used for LSUN scene modeling.

too much RAM to calculate and Google Colab doesn't offer that much RAM all the time. Except the restriction on computing power, other models are not picked for this project because either they focus on greatly different areas, StyleGAN for example [9], or they don't work as well as DCGAN on pictures, such as General GAN [10].

The prepossessing is actually the main focus in this project after the model is chosen. Based on the principle GIGO (Garbage in, garbage out) [11] and the general idea of Machine Learning, the quality and the feature distribution of the input data determine what the model can learn and what the output will be. Typically, Open-CV [12] and Scikit-Image [13] all provide useful functions to deal with images. Since the layout of the smartphones is the feature that this project is aimed to study, the edge detection implementing in traditional computer vision ways will be used in this project. By comparing several edge detection methods, including OpenCV's Canny Edge Detection [14], Scikit-image's Li thresholding [15] , and Scikit-image's Canny Edge Detector [16], there is one implementation of OpenCV's Canny Edge Detection approach that seems to be the most adaptive one [17]. It does not require a parameter and can auto adapt the parameter based on the images and gives a decent result on this smartphones' dataset.

Transfer Learning is also used as one of the techniques in this study. Transfer Learning, based on Stanford's course CS231n's website [18], can be used in three major scenarios and one of those three is pretrained models. This technique, loading pretrained models, will be used in this paper as the dataset of this experiment has a relatively small size and thus the quality of the trained model can not be guaranteed. According to Yonsinski, J. et al. in their paper "how transferable are features in deep neural networks", they suggested that for deep neural models trained on natural images, the first layer's features are not specific for one task or dataset, and the only thing to make sure is that features are ultimately transformed from general to specific by the last layer [19]. Thus, transfer learning is used in this study to load weights of pretrained models (DCGAN models), which have been trained with a larger dataset or which are publicly acceptable, such as those published in Facebook's GAN zoo [20] and those published on PyTorch Hub [21]. After the weights are loaded, all the weights of features are frozen except the last layer's, so that the model can be trained to be specific to this certain task by training the weights on the last layer.

4. Experiments[1]

4.1. Dataset

The dataset is customized and collected from droidchart.com [22]. The website provides detailed information of more than 6500 smartphones as well as the images. A classic python library Selenium [23] for crawler is used. To make the smartphone images more diverse, some old classical mobile phone images are also downloaded from Google and added to the dataset.

Figure 6. Example of Training Images

In addition, because the quality of downloaded images is relatively poor, Super Resolution, a pre-trained model with an efficient sub-pixel convolution layer, is also used [24] to increase the spatial resolution of images.

4.2. Software

Apart from the python library Selenium to collect images for the dataset, PyTorch and TensorFlow are also used to build DCGAN. OpenCV is also used in the image preprocessing session as mentioned above.

4.3. Hardware

This project uses the GPU provided by Google Colab [25].

[1]All codes are available here: `https://github.com/tliu292/A-Smart-Way-to-Design-Future-Smartphones`

4.4. Procedure

This project undergoes many different trials by running DCGAN with various training sets.

In the first trial, the raw dataset with about 6000 smartphone images is used to train a DCGAN model built with TensorFlow and Keras as the backend from scratch [26].

In the second trial, about 800 smartphone images are randomly picked from different years, including dozens of old mobile phone images. Then, the smartphone screens are manually covered up with white rectangles. Next, the Canny Edge Detection is used over these masked images to generate pictures with white lines over the layouts of smartphones and leave the remaining areas with black. The same DCGAN model implemented above is then used to train on this smaller training set.

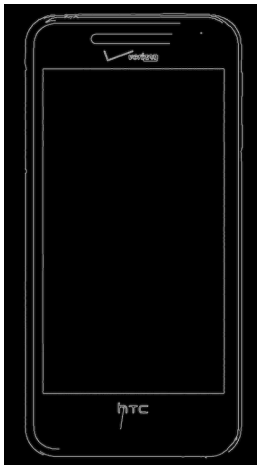

Figure 7. After Edge Detection

In the third trial, transfer learning gets involved. The model is not trained from scratch and instead uses weights trained from the raw dataset in the first trial. The layers' weights are then frozen so that the information retrieved from the previous training set is not lost. However, the last layer's weights are not frozen and they are trained with the smaller dataset used in the second trial again. After training, DCGAN is applied to generated smartphone images. It is interesting to see whether two different kinds of training sets would improve the quality of generated images.

In the fourth trial, instead of using weights trained from the raw dataset, transfer learning uses weights of a pretrained DCGAN model with a larger dataset "FasionGen" [27]. And again, the last layer's weights are not frozen and are trained with the smaller dataset used in the second trial. DCGAN is then used to generage images.

In the last trial, the smartphone images are colorized manually. Around 250 images from the raw dataset are picked and different parts of the smartphones are filled in with different colors. Figure 8 shows an example of smartphone image after colorization. The screen is filled in with green, cameras with blue, buttons on screen with white, buttons on edge with red, and the speaker with yellow. DCGAN is then applied to these colarized images. The idea of coloring is inspired by House-GAN mentioned in the Related Work section.

Figure 8. After Colorization

5. Results and Discussion

In the first trial, the training set is the raw dataset with about 6000 smartphone images. No preprocessing is done. Figure 9 shows the generated smartphone images.

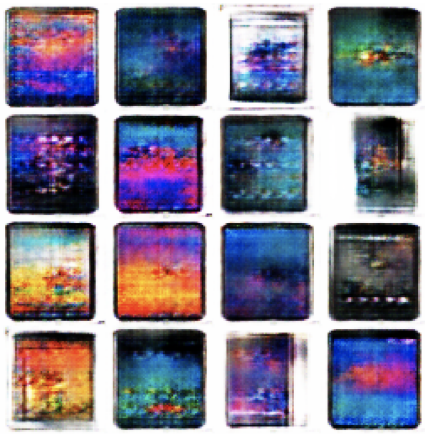

Figure 9. Generated phones - raw dataset as training set

From Figure 9, we can see that DCGAN does not generate satisfying results. The generated smartphones are all covered with colorful wallpapers, and no information regarding the position of microphones, the front cameras, or the buttons are revealed. The most likely reason is that smartphone images in the training set are all covered with

different kinds of wallpapers on their screens, while the screen constitutes a huge part in smartphones' design; thus, the generator and the discriminator in DCGAN are deviated by the colorful screens instead of other small features.

Therefore, in the second trial, we try to manually cover the smartphone screens with white rectangle, hoping that the generator and the discriminator would be not deviated by the wallpaper and would focus more on the small features. In addition, we apply edge detection to the training set such that features such as buttons, cameras, and the overall structure can be more easily detected by DCGAN. However, the number of training set images reduces from 6000 to 800 since manually covering up smartphones with white rectangles takes a long time.

Figure 10. Generated phones - white screen smartphones with edge detection as training set

In Figure 10, the generated smartphones are in black background because all the training smartphones only contain white edges in black background after edge detection, as seen in Figure 7. In addition, the generated smartphones contain features such as the speaker, and the images are no longer deviated from the wallpapers. However, the result is still unsatisfying as the generated smartphones and the training smartphones do not have much difference in appearance.

In the third trial, we try implementing transfer learning to the training set and hope to see something improvement. As mentioned in the proposed method, Weights are first trained on the raw dataset and then frozen except the last layer. The weights in last layer are retrained on the smaller dataset from the second trial.

However, if you look closely at Figure 11, the generated smartphones are even worse than the previous result. For instance, the overall phone structure is much harder to see. The camera, the microphone, and other small features also

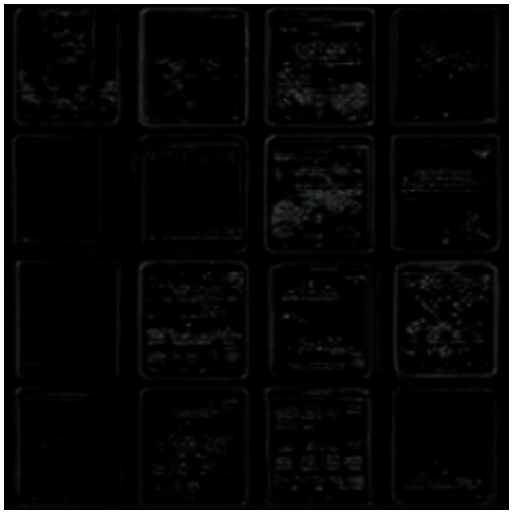

Figure 11. Generated phones - transfer learning with raw dataset

disappear. The screen also gets much dirtier than before. The most likely reason is that DCGAN does not perform well on the raw dataset, as the wallpaper on smartphones' screen gets relatively blurry in the first trial.

Since transfer learning does not work well on small training set, we try transfer learning with a much larger dataset called FashionGen in the fourth trial, but the result is also not good as one can barely see anything from the generated images, referenced in Figure 12.

Figure 12. Generated phones - transfer learning with FashionGen dataset

Thus, we conclude that transfer learning is not a good approach for generating smartphones; then, we switch our gears away from transfer learning and consider colorizing the phones instead in the final trial. The phones are colorized as seen in Figure 8. However, only 250 smartphone

images are used as training set because manually colorizing smartphone features with different colors takes even longer time.

Figure 13. Generated phones - colorization

From Figure 13, the generated smartphones are still a little unclear but they have much better quality than the smartphones generated in transfer learning. Besides long-shaped smartphones, DCGAN also generates some square phones and the reason is still unknown. In addition, DCGAN captures some colored features. If you look closely, most generated smartphones include the green screen, the yellow speaker, the white buttons on screen, and the red buttons on edge. Though some small smartphone features are captured, DCGAN does not generate creative phone designs, as we expected from the Chair Project [4].

6. Conclusions

We have tried various kinds of training sets, including the raw dataset with 6000 images, the dataset excluding phone screens with 800 images, transfer learning with raw dataset and the FashionGen dataset, and colorized phones with 250 images. However, DCGAN does not generate new phone designs under these approaches.

We consider various reasons why the generator cannot produce create phone designs. First, in the Chair Project [4], the original training set contains different kinds of chairs, and the chair design is very different across different designers. However, for smartphones, though different companies produce different smartphone designs, the overall appearance of phones looks very similar, since the screen occupies about 80 percent of the entire product. Because of the similarity in design among various smartphones in the training set, DCGAN is unable to generate new designs like it did in the Chair Project [4].

Second, DCGAN is better at generating images with different textures. Thus, the generator is good at generating images like chairs and human faces. However, smartphones do not contain any textures; and the most distinctive features on smartphones are the shapes.

Thirdly, the transfer learning in DCGAN is better at generating human faces instead of smartphones, which is a primary reason why transfer learning performs so poorly in this project. These is a huge domain gap between the two areas.

Lastly, the training set is too small. The raw dataset only contains about 6000 images, and the datasets that require manual actions are even smaller than the raw dataset. For instance, only 800 images for white-screen smartphones and 250 images for colorized smartphones are trained. The small training set does not give DCGAN enough training space to generate creative designs.

For future directions, this project only tries a single generative adversarial network, DCGAN, for generating smartphone images. There are various other GANs like CGAN, CCGAN, CycleGAN, and etc that we can try. In addition, it would be interesting to see if larger training set can generate better results. Lastly, due to the unsatisfied result in generating smartphones within this project, instead of focusing only on smartphones, we can try various other products, like computers and televisions, when generating images through Conditional GAN. It might generate mixed phones or other weird digital products that would be interesting to see.

7. Acknowledgements

The main part of the dataset is collected from droidchart.com. They may or may not acknowledge our usage of their images; but since this is a class project and is not likely to be published or used for commercial purposes, it should be fine. Additionally, we would love to acknowledge general help and useful suggestions from our professor Sebastian Raschka and our TA Zhongjie Yu.

8. Contributions

Kyle writes the crawler script that collects smartphone images as raw dataset. Tz-Ruei and Zhilin pre-processes the images; for instance, they replace screens with white rectangles or colorize the smartphone features with different colors. Then, for the experiment, Kyle prepares the interface for DCGAN and all three team members run DCGAN over different training sets. For the presentation, Zhilin prepares the introduction and the first trial; Kyle prepares the second trial that involves removing the white screen and the third trial that involves transfer learning; Tz-Ruei prepares the fourth trial that involves colorized phones and also conclusion and future directions. For the paper, each member writes the paper according to what they prepared in the pre-

sentation.

References

[1] C. Grazia, C. Nicoletta, and B. Riccardo, "Innovation and competition in the smartphone industry: Is there a dominant design?" *Telecommunications Policy, Volume 39, Issues 3–4, pp. 162-175*, 2015.

[2] "Smartphone innovations of the 2010s (infographic)," 2020, (Accessed: 29 Feb 2020). [Online]. Available: https://www.helpsmartphone.com/en/articles-privacy-smartphone-innovations-2010s

[3] H. Catherine, "Future mobile phones: what's coming our way?" *Uswitch mobiles*, 20109.

[4] P. Schmitt, "The chair project," *https://philippschmitt.com/work/chair*, 2019.

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 11 2015.

[6] S. Radhakrishnan, V. Bharadwaj, V. Manjunath, and R. Srinath, "Creative intelligence – automating car design studio with generative adversarial networks (gan)," in *Machine Learning and Knowledge Extraction*, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds. Cham: Springer International Publishing, 2018, pp. 160–175.

[7] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation," *eprint arXiv:2003.06988*, 2020.

[8] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," 2017.

[9] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00453

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[11] "Garbage in, garbage out," Apr 2020. [Online]. Available: https://en.wikipedia.org/wiki/Garbage_in,_garbage_out

[12] "Opencv," Apr 2020. [Online]. Available: https://opencv.org/

[13] "News¶," *scikit*. [Online]. Available: https://scikit-image.org/

[14] "Canny edge detection." [Online]. Available: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html

[15] "Li thresholding¶." [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/developers/plot_threshold_li.html#sphx-glr-auto-examples-developers-plot-threshold-li-py

[16] "Canny edge detector¶." [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/edges/plot_canny.html#sphx-glr-auto-examples-edges-plot-canny-py

[17] R. Jansekok, A. Rosebrock, Michele, Y. Daoust, Tyler, Bosmart, T. Clemans, Brian, David, Chuck, and et al., "Zero-parameter, automatic canny edge detection with python and opencv," Apr 2020. [Online]. Available: https://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/

[18] Apr 2020. [Online]. Available: https://cs231n.github.io/transfer-learning/

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" 2014.

[20] Facebookresearch, "Pytorch gan zoo," 2020. [Online]. Available: https://github.com/facebookresearch/pytorch_GAN_zoo

[21] Pytorch, "pytorch/hub," Apr 2020. [Online]. Available: https://github.com/pytorch/hub

[22] "Smartphone specifications, pictures, videos and comparisons." [Online]. Available: https://droidchart.com/en/

[23] "Selenium with python¶." [Online]. Available: https://selenium-python.readthedocs.io/

[24] Pytorch, "pytorch/examples," Jan 2019. [Online]. Available: https://github.com/pytorch/examples/tree/master/super_resolution

[25] "Google colaboratory." [Online]. Available: https://colab.research.google.com/notebooks/gpu.ipynb

[26] R. Agarwal, "An end to end introduction to gans," Dec 2019. [Online]. Available: https://towardsdatascience.com/an-end-to-end-introduction-to-gans-bf253f1fa52f

[27] "Fashiongen challenge." [Online]. Available: https://fashion-gen.com/