# **1. Exploring Face Detection for the Real and Virtual World - Devon Urlich, Jeremy Chen, Victor Chu**

This project explored face detection on cartoon characters. They started with a baseline facial detection algorithm called MTCNN and tested it on a 50/50 dataset of real and cartoon images. They also used a custom cartoon facial detection system: a sliding box window with a binary classification system for each possible window. This model struggled with some cartoon characters because of strangely shaped faces, like tall, narrow faces or short, wide faces. Finally, they combined these two models and tested them on the 50/50 dataset, appending the sets of bounding boxes together. This resulted in many false-positive results and an average precision of only 36%. Their solution to this problem was to introduce a binary classifier before passing the image to either MTCNN or their custom classifier. This classifier would output “cartoon” or “real world,” and would pass the image to either MTCNN (if the image was classified as real) or to their custom classifier (if the image was classified as a cartoon). This classifier had > 90% accuracy predicting real vs. cartoon, and their final detection system had a 63% average precision. In the future, this model could be further improved by focusing on the quality of the dataset. The dataset they used, while convenient for this project, had some images with text over the top of the faces. This made the detection system predict no face where there actually was a face, either real or cartoon. A future project could be removing the text from these images or finding new images without covering text.

**2. Using Image Features to Defend Against Adversarial Attacks - Mason El-Habr, William Makinen**

This project focused on defending against attacks against feature extraction models. It focused on 4 different attacks: A noise attack, a Fast Gradient Sign Attack (FGSA), a Carlini-Wagner L2 Attack (CWLA), and the Deepfool attack. These attacks all use different mathematical equations to add noise to an image, which is imperceptible to the human eye. Previous solutions to this problem have been adding adversarial images to the training dataset, but this is difficult because it needs a set of adverse images and it can only be trained on known attacks (and therefore will not be robust against new, unknown attacks). This group tried to solve this problem by combining convolutions and feature extraction on both the RGB and greyscale versions of the image, in two different types of models. The first model was a linear concatenation of the features extracted from the RGB and greyscale images, and the second was a parallel calculation. The former extracted features from the greyscale image and concatenated them onto the results of convolving the RGB image multiple times, while the latter also convolved the features of the greyscale images. These methods were mostly unsuccessful, but they did make some attacks work harder. In particular, the Deepfool attack struggled with changing the accuracy when the model extracted HOG features from the images. The magnitude of this struggle was measured using the magnitude of perturbation. FGSA and CWLA were the most robust against their model: that is, these attacks had the lowest magnitudes of perturbation and their runtimes were affected the least.

**3. Variable Image Colorization - Jens Clausen & Mark Abramowitz**

This project explored the colorization of greyscale images. This is a difficult task, mainly due to the fact that the model must go from smaller to larger data (for example, the input might be NxNx1, and the output would have to be NxNx3). They tried two approaches to this solution: the first was building their own model using the U-Net architecture, and the second was using DeepAI’s model to evaluate different grayscaling techniques. The first approach caused a lot of problems because of the difficulty of the problem at hand: this group struggled with training their model on a single image (for which the model should always perfectly recolor, given that the only “label” it encounters is the correctly colored image). This required a pivot to using DeepAI’s model so that they could still do part 2 of their project, which was evaluating different greyscaling techniques using the colorization model. To evaluate these techniques, the group first converted a color image to greyscale and then colorized it using DeepAI’s colorization model. They evaluated each greyscaling technique based on loss (from the original color image to the recolored image), vibrancy, and saturation. They found that most greyscaling techniques lost much of the vibrancy of the original image, likely due to their implementations. This is because many greyscaling techniques average rgb values in some form or fashion, which means that much of the information that goes towards making a picture vibrant is lost and cannot be reconstructed by the colorizer. In future work, this group wants to add a human component to training greyscaling techniques: that is, give more weight to grayscale images that result in a more vibrant recolored image (as determined by human labelers) than those that do not.