A Deep Learning Solution to Catfish Detection

Kris Kaya (kkaya23) & Stefan Swaans (sswaans)
Mentor: Patrick Cho
Department of Computer Science, Stanford University

June 11, 2018

Abstract

The increase in false profiles on social media has resulted in users being lured into relationships by a fictional online person, a phenomenon known as catfishing. For our project, we implemented a facial recognition algorithm that when combined with a pose recognition algorithm, provides a model for detecting false social media profiles. Performance of two architectures was compared: a standard Convolutional Neural Network and a Transfer Learning Model using the Inception-Resnet v2 model pre-trained on the ImageNet dataset. We found that the transfer learning model outperformed the standard CNN, which makes us optimistic about the use of transfer learning as a strategy to implement an algorithm able to detect false profiles.

1 Introduction

The rise of online dating platforms has made the task of meeting potential partners easier, as users can connect with others within a radius of several miles. However, this has resulted in some using such services for reasons beyond their original intentions. The ease of creating a profile for using applications like Tinder as well as the myriad of available online images has resulted in a phenomena known as catfishing, defined as "luring someone into a relationship by means of a fictional online persona". These false profiles have led to more instances of catfishing, as there is no disincentive to creating false profiles for the purposes of practical joke or solicitation. We Interestingly, the average user of such services can usually tell if a profile is legitimate or illegitimate. We hope propose a deep learning solution to help decrease instances of catfishing on online dating platforms. Our input is a greyscale image and we hope to train two models to accurately recognizes faces in a database and output a predicted individual based on the image.

2 Related Work

After scanning the literature, there were very few previous studies that analyzed facial recognition strictly for catfish detection. However, there is literature surrounding general facial recognition strategies. Phillips et al. outline the FERET Evaluation Methodology for Face-Recognition algorithms, which makes it possible to independently evaluate facial recognition algorithms. It represented the de facto standard for measuring the performance of facial recognition algorithms and shows the factors effecting performance. In 2002, Flynn et al. described the Face Recognition Grand Challenge, which was a six-experiment challenge problem with data corpus of 50,000 images. These problems were meant to incentivize researchers to achieve reduced error rates in facial recognition systems. The two above papers framed various initiatives that prompted an increase in facial recognition capacity. Maseem et al. present an approach to facial identification using linear regression. Using a nearest subspace classification algorithm, a linear model representing an image was created and the least-squares method was used to generate a reconstruction error that was to be minimized. Vasilescu and Terzopolous apply multilinear algebra to obtain a representation of facial images which separates factors such as poses and lighting. This model was able to successfully improve facial recognition rates. Finally, Wiskott et al. presented a system of recognizing human faces from single images out of a large database with one image per person. Their model extracts concise face descriptions in the form of image graphs, which describes facial points by

sets of wavelet components. These image graphs are compared for recognition. We analyzed these techniques to gain intuitions about our models as well as how to use existing techniques in the literature to address our problem.

3 Dataset and Features

We used the Yale Faces Data Set, a data set compiled by scholars at Yale University. The dataset contains 165 grayscale images of 15 individuals (14 males, one female). Each subject has 11 different images, each with a different facial expression: center-light, wearing glasses, left-light, happy, without glasses, normal, right-light, sad, sleepy, surprised, and winking. We picked this dataset because it's size made it easy to iterate through, which allowed us to focus on other optimizations. It also is a readily accessible dataset which made it easy to collect. Moreover, since there are a wide variety of facial expressions, it would require the model to determine that different faces can correspond to the same person and therefore, makes the model more robust. Since the data set was small, we did approximately a 60-20-20 split for the train, dev, and test sets. We wanted to ensure that every person had an image in the train, dev, and test sets. However, we also wanted to ensure that which facial expression/configuration appeared in which of the three sets was randomized. Therefore, we shuffled each of the 11 images associated with a profile. Then, we put the first seven in the training set, the next two in the dev set, and the final two in the test set. This made sure that we didn't have disjointed dev and test set performance as a result of the two sets having a different distribution.

Our raw input features were the pixels of each grayscale image, which were 256x256 in the basic CNN and 128x128 in the transfer learning solution. Additionally, we preprocessed images in the transfer learning solution, primarily through reflections and noise addition to make the model more robust. We did not have any derived input features. They are appropriate for this task because they are standard inputs for a facial recognition model.

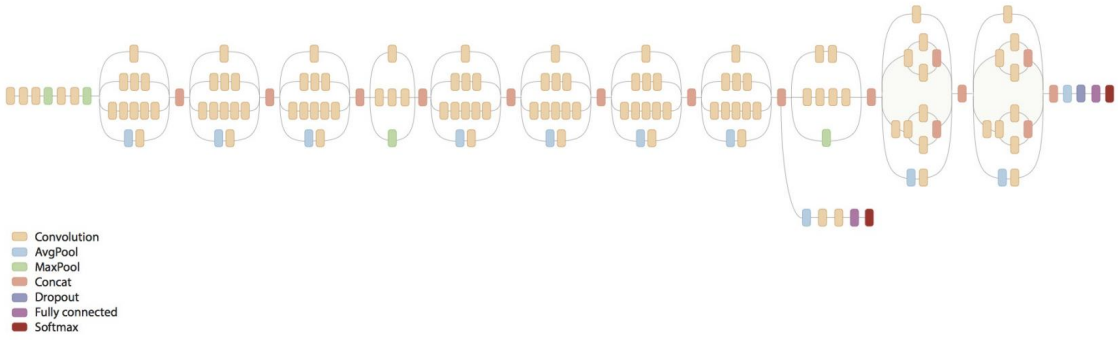


Figure 1: Architecture of the Inception-Resnet v2

4 Methods

We tested the performance of two models to determine which one performed best. The first model was a base convolutional neural network with the following architecture: 3x3 CNN $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ 2x2 MaxPool network. The 3x3 CNN, or Convolutional Neural Network, passes a specialized 3x3 filter over the greyscale image in order to detect edges. The result of this filter pass is an image in which areas of high contrast have been highlighted (given large pixel values). The BatchNorm layer normalizes the mean and variance of the output layer. A normalized output layer neutralizes huge differences in scale between input features, reducing oscillations in gradient descent, which speeds up learning. The 2x2 MaxPool layer takes the maximum value in each 2x2 section of the matrix and saves it into a new matrix. This has the effect of condensing the input while not losing information about important features, as the maximum value usually

corresponds to the features the network is looking for. We used Adam Optimization on our model to improve performance, as well as a Dropout layer to prevent overfitting. Dropout is a regularization technique that randomly turns off hidden layer units within a neural network. Adam is an optimization algorithm that combines momentum and RMS Prop that helps an algorithm converge to an optimal solution faster. We used the model to classify between the 15 different people and used a sparse softmax cross entropy loss function.

The second model we tested was a transfer learning solution. We used a research network by Google, Inception-Resnet-v2 (shown in Figure 1), pretrained on the ImageNet dataset. We then trained it on our own dataset and used it to classify between the 15 different people. We again used a sparse softmax cross entropy loss function. Explained simply, the softmax cross entropy output layer allocates a relative probability for each predefined classification type, then chooses the type with the highest relative probability. It calculates these probabilities with the following loss function: $J(\Theta) = -\frac{1}{n}\sum_i^n [y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i})]$.

Our approach to combating catfishing is to use various types of images as a "key" to a profile. When a new profile is created, the user would be asked to take several pictures in given poses, each with their face visible. A completed version of this network would then analyze those four to five images and ensure that the expected pose is correct in each, as well as that the face is consistent across all images. These images act as proof of the user's identity, as it would be difficult for a catfisher to find images of someone else with the required poses. This would raise the barrier to the creation of false profiles immensely. Our ideal model, then, has two components: facial recognition and bodily orientation/pose recognition. The first task would be a facial recognition one, as we want to confirm that the face is the same across all given pictures. If it isn't, then we know that the user is attempting to use someone else's face to make a false profile and we would therefore reject it. The second component of our algorithm is a pose recognition task. We would isolate 5 poses chosen at random from a set of trained poses to operate as a "password" to a given profile. A user creating a new profile will be asked to upload pictures in these 5 poses to determine if they are legitimate. If a given image is not of the correct pose, we know that the person in the image is not the one creating the profile and the profile is most likely false. Given the time constraints of the quarter, we decided to successfully implement the facial recognition component of our algorithm to train a neural network to recognize faces regardless of the expression.

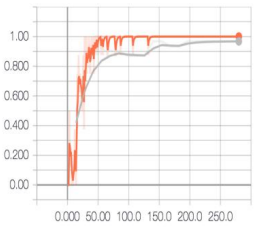
5 Experiments/Results/Discussion

We systematically tuned the following hyperparameters: image size, momentum term, learning rate, and batch size. After testing various values we concluded that the hyperparameters below combined to create the best accuracy and lowest loss.

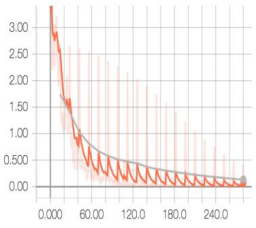
α	1e-2.5
β	0.7
Image Size	256
Batch Size	8

Moreover, when running the transfer learning model, we noticed that there were certain epochs where our cost and accuracy failed to converge at an optimal solution. Therefore, on the model, we implemented learning rate decay in attempt to converge towards a more optimal solution, which was ultimately successful. This is evidenced by the graphs illustrating model 2 learning rate and model 2 training loss

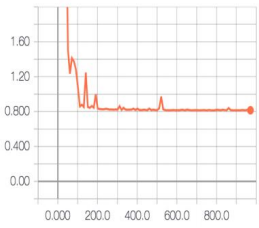
Pictured below are the Tensorboard graphs depicting results from Model 1 (our basic CNN model) as well as Model 2 (the transfer learning Inception-Resnet-v2 model):



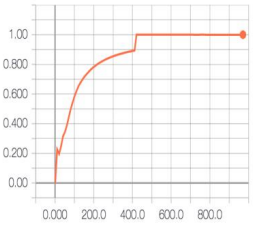
(a) Model 1 Train/Dev Accuracy



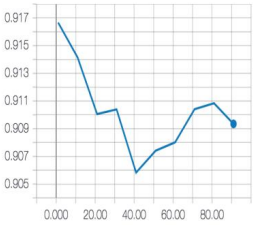
(b) Model 1 Train/Dev Loss



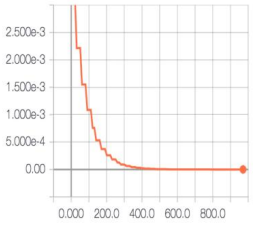
(c) Model 2 Train Loss



(d) Model 2 Train Accuracy



(e) Model 2 Test Accuracy



(f) Model 2 Learning Rate

Overall, we were satisfied with the success of our models, given that both achieved approximately 90% test accuracy. However, the Inception-Resnet v2 transfer learning model achieved greater test accuracy with an accuracy of 91.09%. We experienced a small amount of overfitting, as our train and dev sets commonly achieved between 96% and 100% accuracy. To combat this, we implemented AdamOptimization and Dropout. However, the dataset we used was small enough that with enough epochs, the model could reach 100% train accuracy reliably. We believe that, with a larger dataset, we would likely have experienced a lesser differential between our train/dev accuracy and our test accuracy.

After doing error analysis, we recognized that there were some issues with our model. First, in our Model 1 train/dev loss graph, we noticed that there was some noise in the plot. We realized that this was the result of not shuffling our mini-batches as we were iterating, which contributed to the loss sometimes increasing because our mini-batches may have sometimes included images only from a single person. Additionally, we recognized that there were certain expressions that our model struggled with, for example the right light expression (shown in Figure 2). We suspected that this was a result of our model partial learning the background of images as a heuristic for identification. We attempted to implement Occlusion in attempt to visualize what our model was identifying but were unable to get it working. Also, we found that initially, Model 2 was performing incredibly poorly. We found this was because the model was preprocessing our dataset to make it bigger by cropping and rotating. We noted that this was unhelpful as for our algorithm, it is unlikely it would need to identify a face rotated beyond 30 degrees. After removing this preprocessing, our model performed much better.

Figure 2: The right light expression, a commonly misclassified expression

While our models did achieve a reasonable amount of accuracy, we recognize that part of this was the result of our relatively small dataset. We chose to work with such a dataset such that we

could iterate quicker and focus more of our efforts on optimization of our algorithm. Moreover, since we tested two different models, each model required us to manipulate our data such that it was a valid input, which would have been substantially more time consuming with a larger dataset.

6 Conclusions/Future Work

Between the two models, we found that applying transfer learning via the Inception-Resnet v2 model was the better performing model. We were not necessarily surprised by the result, as the problem represented a good case for transfer learning - namely when there are more amounts of preexisting images A that we can apply to learn a smaller data set B.

We proposed that the solution to the problem would involve both a facial recognition and a pose recognition component to detect whether or not a profile was legitimate. Had we had more time and computational resources, we first would have trained our model on a much larger dataset. The time that our model required to train, especially for the transfer learning model, would have most taken several days to train with a larger dataset. Additionally, we would have had to reformat the data, which would either require a computationally expensive program or a significant amount of time. Since we focused on successfully implementing the facial recognition component of our solution, the next step we would take is to implement the pose recognition portion of the model. We would work to optimize performance of a neural network meant to classify between a variety of poses. An interesting thing to note is that given the success of the Transfer Learning solution, we are optimistic about the feasibility of a solution for the problem of catfishing. A significant amount of literature exists on the tasks of facial recognition and pose recognition independently. Therefore, transfer learning presents a possible strategy for combining the task of pose and facial recognition to be solved by a single neural network or two neural networks in parallel. We believe this model would be sufficient to combat the problem of catfishing and to make online dating services more secure.

7 Contributions

We believe both team members contributed equally to the project as both were present at every mentor meeting and were together whenever the project was being worked on. Stefan formatted the dataset, manipulated the starter code and the Inception-Resnet v2, and ran the model and tabulated results. Kris formatted the poster and report and helped in optimizing the algorithm through hyperparameter tuning and the transfer learning strategy.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] I. Naseem, R. Togneri and M. Bennamoun, "Linear Regression for Face Recognition," inIEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 11, pp. 2106-2112, Nov. 2010

[3] Phillips, P. Jonathon et al. “Overview of the face recognition grand challenge.”2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)1 (2005): 947-954 vol. 1.

[4] P. J. Phillips, Hyeonjoon Moon, S. A. Rizvi and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 10, pp. 1090-1104, Oct 2000.

[5] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear image analysis for facial recognition," Object recognition supported by user interaction for service robots, 2002, pp. 511-514 vol.2. doi: 10.1109/ICPR.2002.1048350

[6] Laurenz Wiskott , Jean-Marc Fellous , Norbert Krüger , Christopher von der Malsburg, Face Recognition by Elastic Bunch Graph Matching, IEEE Transactions on Pattern Analysis and Machine Intelligence, v.19 n.7, p.775-779, July 1997