

Fake Face Recognition Using Images Created From FakeAPP

I. Definition

Project Overview

Deepfake, is a deep learning based framework which employs human image synthesis technique that enables users to swap human faces. Mainly, it is used to combine existing images and videos onto source images or videos in a very realistic manner.

According to sources, Deepfakes like FakeApp and MyFakeApp may be used to create fake celebrity pornographic videos, revenge porn, fake news and malicious hoaxes [1]. Pornography created by Deepfake, particularly on sites like Reddit Twitter and PornHub [2-4] has been banned.



Figure 1 – Nicolas Cage FakeApp

In the picture above, the original Nicolas Cage is on the left and the fake one is in the middle. The screenshot was taken from a deepfakes video from youtube. It is obvious that the similarity between the two persons is very high.

Problem Statement

Near real-time counterfeiting of facial expressions in existing 2D video can cause major issues to the individuals displayed in the videos. The main goal is to create a classifier system running on a desktop initially which is capable of recognizing fake images of individuals.

I propose an image classifier system using CNN where a user takes a screenshot from a video that suspects it is fake and feeds it to the system. Then a prediction followed by a probability estimation is produced suggesting the user regarding the authenticity of the picture.

The potential tasks involved for the construction of such application are the following:

1. Download FakeApp application.
2. Create from dataset from the two videos
3. Choose a pre-trained CNN architecture
4. Use transfer learning to train an image classifier based on CNN architecture
5. Classifier evaluation

The final application is expected to be useful for users where they are victims of fake video creations.

Metrics

Binary classification involves classifying the data into two distinct groups, for example in our case we need to know whether a face is fake or not. The formal representation of a binary prediction is depicted in equation 1 below:

$$\hat{y} \in [0,1]$$

Equation 1 - Binary Classification

Also, binary classification estimators produce a probability of a target variable to be 0/1. Thus, to evaluate such a model a confusion matrix is used and with its help we can calculate several performance metrics. In this case, since the performance metric evaluated in the benchmark model is accuracy, I am going to evaluate the same metric as a performance indicator [5]. Accuracy is depicted in equation 2 below and is the division of correct predictions divided by all the observations of the dataset:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

II. Analysis

Exploratory Visualization

In this study, the biggest part of the dataset is consisted of images created both from original and fake videos of Nicolas Cage. Also, a smaller part the fake images is extracted from public repositories like the google and Pinterest. A sample is depicted in table

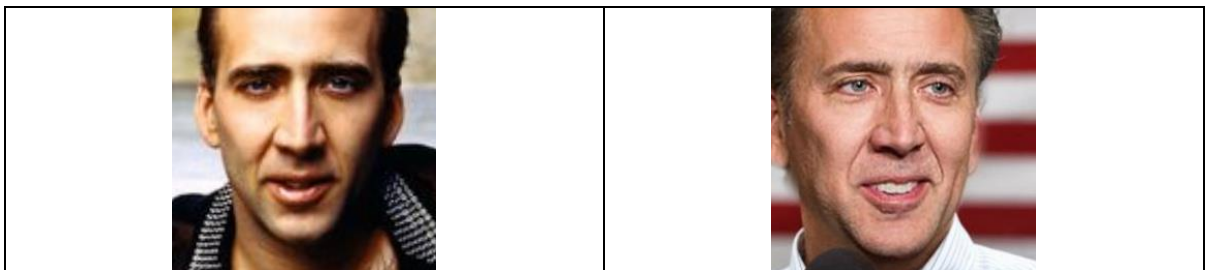


Table 1 - Sample from the dataset extracted

The dataset is consisted of 169 fake images and 367 original images. Although FakeApp extracted a lot more frames from the videos, many of them were almost identical and I was worried about the robustness of the model. To that end, I chose to select only this specific number of images were in my opinion reflect Nicolas Cage's original and fake face features. The images are aligned into different dimensions, but the input size of the chosen architecture is 224x224x3 RGB format.

I've converted the labels of the images into one-hot encoding e.g. [0,1] were, 1 reflects the authenticity of the image and 0 implies that the image is fake.

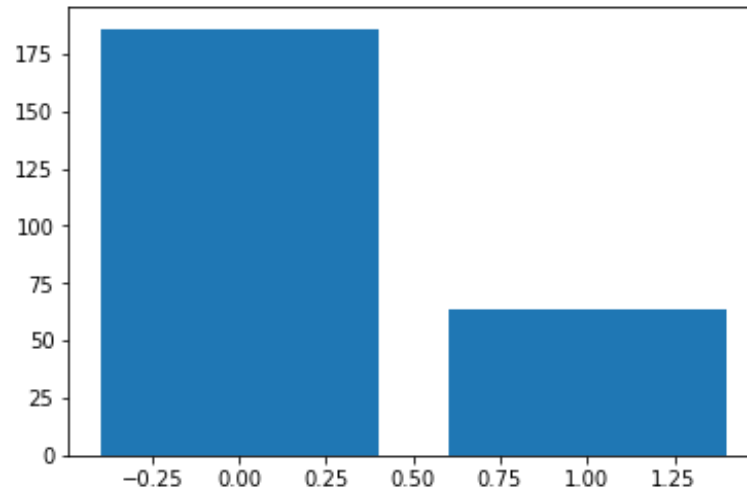


Figure 2 - Image Label Distribution

Labels distribution depicted in figure 2 above, clearly shows class imbalance in the data. At this point recall should probably be more robust measure given the uneven distribution of the data. Despite that, accuracy is the metric used for model evaluation since in the future it is expected more fake data will be fed to the system.

Algorithms and Techniques

Literature review reveals that CNN yields extremely good results compared to other image recognition algorithms [6]. Convolutional Neural Networks are used for computer vision problems like image classification tasks. Also, forms the basis for other computer vision tasks such as localization, detection, and segmentation [7]. The main structural element of a CNN architecture is the convolutional layers. Convolutional layers serve as feature extractors, in order to learn the feature representations of the input images. More information about CNN one can find in these papers [8].

For the purposes of this research I will apply three CNN architectures, one based on VGG-16, one LeNet-like and one custom, implemented for the purposes of the project I submitted for the Deep Learning section of this nanodegree. The first is a pre-trained model which received training on LFW image dataset for face recognition and classification [9].

VGG_Face is a pre-trained model based on celebrity's names dataset [9]. Given the fact that the dataset is similar to the one used for training and very small I will try to adjust the weights by cutting the end of the convnet and perform fine-tuning.

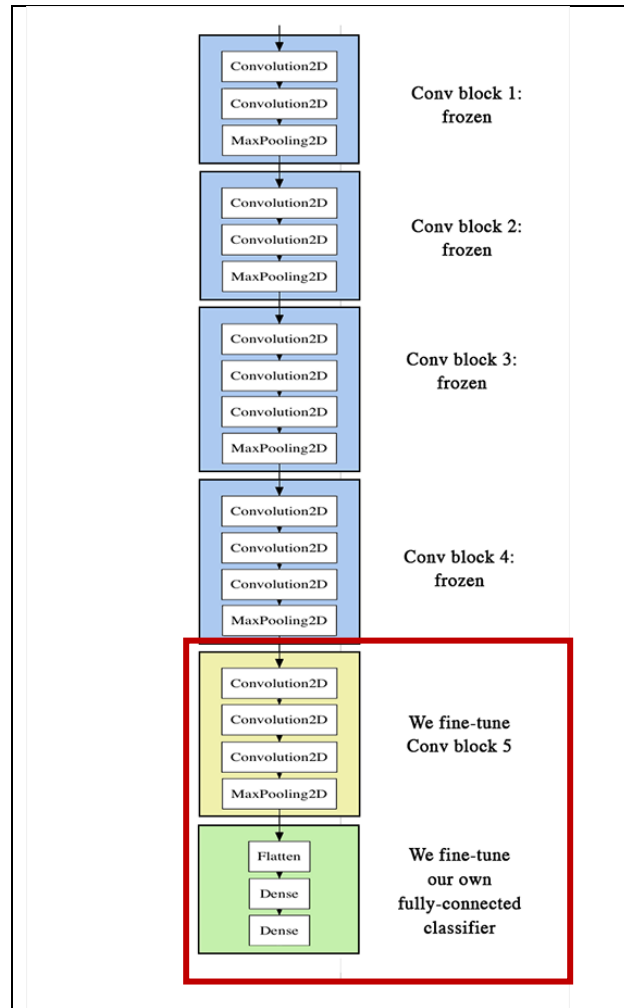








Table 2 - VGG-16 Architecture [10]

Table 2 above, depicts the architecture of VGG-16 model. In the red square are the parts which I will use to can use to fine tune the convnet.

III. Benchmark Model

Previous research on face matching-recognition based on FaceNet framework yielded very good accuracy results in celebrity face recognition tasks [11] utilizing MS-Celeb-1M and CASIA WebFace databases. More specifically, the pre-trained model Inception ResNet v1 yielded over 98.0% accuracy [12].

Although the results recognizing celebrities faces where remarkable, the model failed to distinguish fake faces created from FakeApp. In more detail, face matching failed to separate the original from the fake face of Nicolas Cage. He figures below present the respected results:

	
Distance = 1.00 Result = Same Person	
	
Distance = 1.00 Result = Same Person	
	
Distance = 0.99 Result = Same Person	





	
Distance = 0.95 Result = Same Person	
	
Distance = 0.82 Result = Same Person	

Table 3 - Nicolas Cage Face Match

At the same time, a VGG-16 based model (vgg_face) is fine-tuned using the LFW dataset in order to predict a celebrity's name. I fed the same pictures as above in order to check if the model is going to distinguish the original from the fakes pictures of Nicolas Cage, however the model failed to classify both original and the fake ones.

As a result, these two pre-trained models are going to be the benchmark models to which my implementation is going to compare with. Finally, the constructed label vector is used to calculate the training and validation error during training and validation process.

IV. Methodology

1.1.Data Preprocessing Steps

Below the process of acquiring and preprocessing the entire dataset is presented:

Extraction

The dataset is created from two kinds of images. Firstly, from videos depicting authentic and fake pictures of Nicolas Cage and images from public repositories like Google Image and Pinterest.

Image Alignment

The first step was to align the images extracted from FakeApp and other repositories. Alignment is referred to the process of locating a face and cropping an image. The purpose of that process is to minimize the redundant information and focus only in the face features of an image [13]. FakeApp provides natively face alignment, hence I used it to that end.

Image Rescale/Resize/RGB to Gray Scale

Although the initial plan was to use the original dataset and just rescale it by dividing all the pixels of an image with 255. However, given the structure of the images is computationally expensive to train any model with the current hardware as mentioned in the previous section. Thus, o I given the limitations of the hardware the images were resized into 96x96x1 and converted into gray scale format.

Data Augmentation

Unfortunately, the dataset was very small compared to other datasets used in deep learning implementations. The immediate danger from that was overfitting the data since the number of the images is very small. To mediate that issue I used data augmentation approach to create diversity. More details on the ImageGenerator configuration are given in the notebook.

1.2.Implementation

The implementation is divided in two parts, the first part includes the construction and training of a simple CNN architecture that is going to be used as a benchmark too. The architecture of the custom implementation is depicted in figure

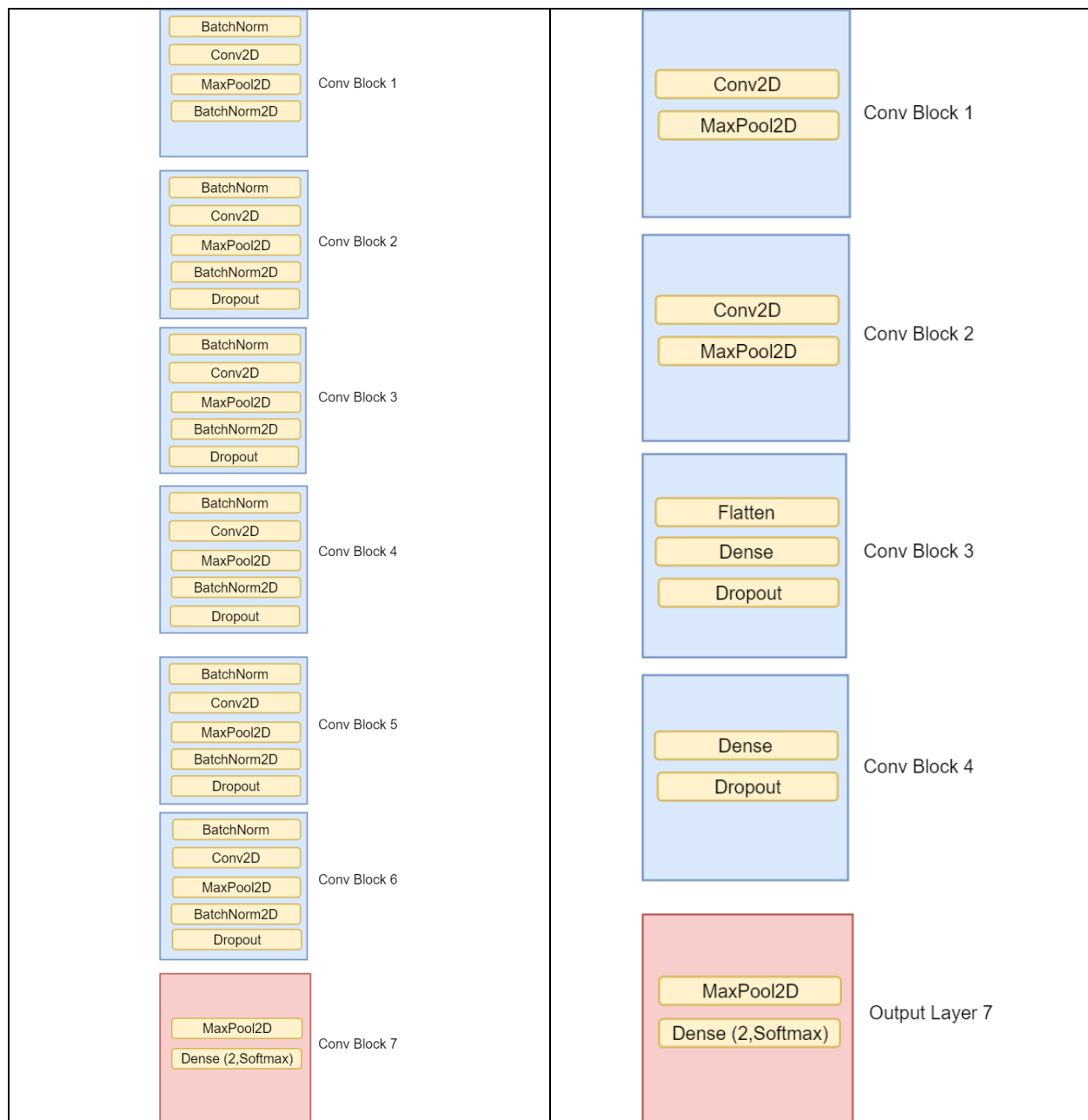


Figure 3 – Custom and LeNet-Like CNN Architecture

The second part, includes the fine-tuning of the acquired VGG-16 based model described in the previous section. In more detail, I removed the last layer and the two fully connected layers and I added a batch normalization, a dropout, a global average

pooling and a dense layer. Finally, I used a LeNet-like model architecture [14] in order to increase the chances to get a better classification results.

Fine-tuning VGG-16 model using the same amount and preprocessing approach for the data took a lot more than expected. In more detail, training the model given the following parameters, *epochs* = 60, *batch_size* = 20, *rmsprop*(0.00001) and *loss* = *categorical_crossentropy*, took 2:15:40 just for one epochs and running on GPU.

To mediate the time performance problem, I decided to resize and convert the images from RGB to grayscale format. That yielded more issues because many pre-trained models were trained in colorful dataset like ImageNet. Thus, I decided to stop the training of VGG-Face model and keep the custom implementation and LeNet-Like models as my primary models model.

V. Results

Model Evaluation and Validation

During the research for the architecture that fits the needs of that project, I implemented and tested several versions of a number of models. Presenting visualizations of them would be a tedious task. Hence, I decided to present the results of the models that yielded the top performances.

Although the data amount is very small, data augmentation did not seem to help a lot. It is important to mention that augmentation was performed only on training data and not on validation and test data.

1. Custom Model

The performance using the data with and without any form of augmentation are presented in table 4 and 5 accordingly. The accuracy using the following hyper-parameters, *batch_size* = 5, *epochs* = 100 and *loss* = *rmsprop* (0.0001) reached 82.31%. On the other hand, accuracy with the same hyper-parameter configuration using the augmented data reached 86.51%

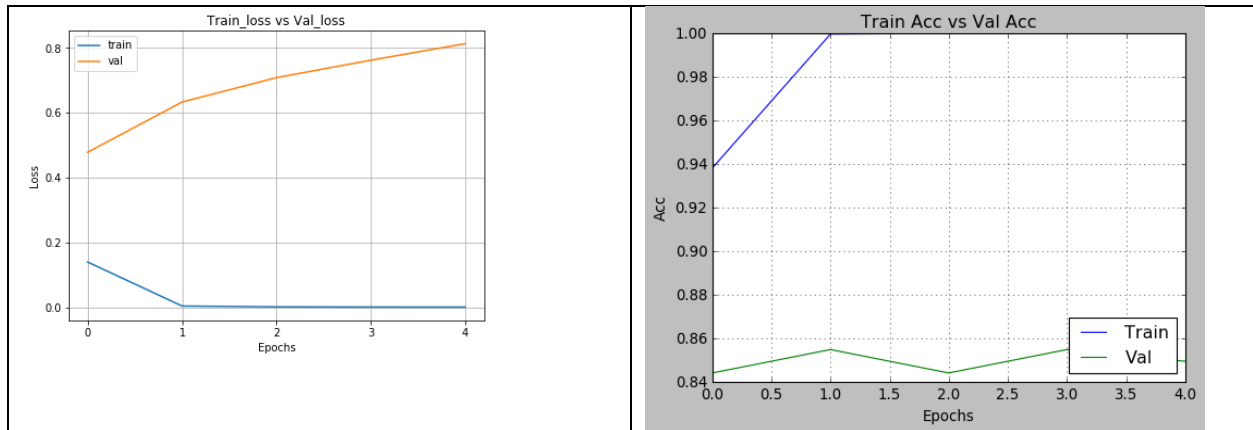


Table 4 - Custom ConvNet Performance during Training

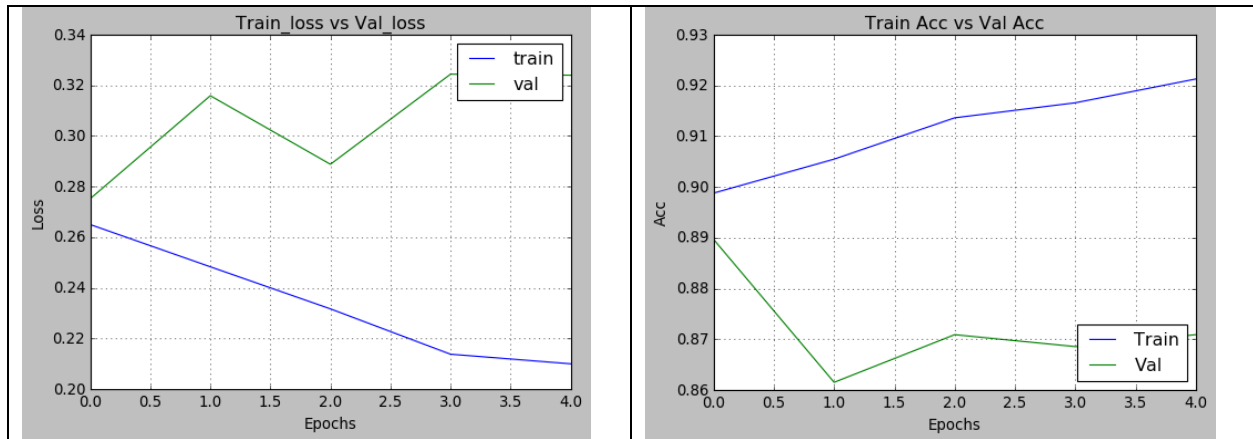


Table 5 - Custom ConvNet Performance during Training using Augmented Data

2. LeNet-Like Model

The performance using the data with and without any form of augmentation are presented in table 5 and 6 accordingly. The accuracy using the following hyper-parameters $batch_size = 10$, $epochs = 100$ and $loss = rmsprop (0.0001)$ reached 82.95%. On the other hand, accuracy with the same hyper-parameter configuration reached 79.87%

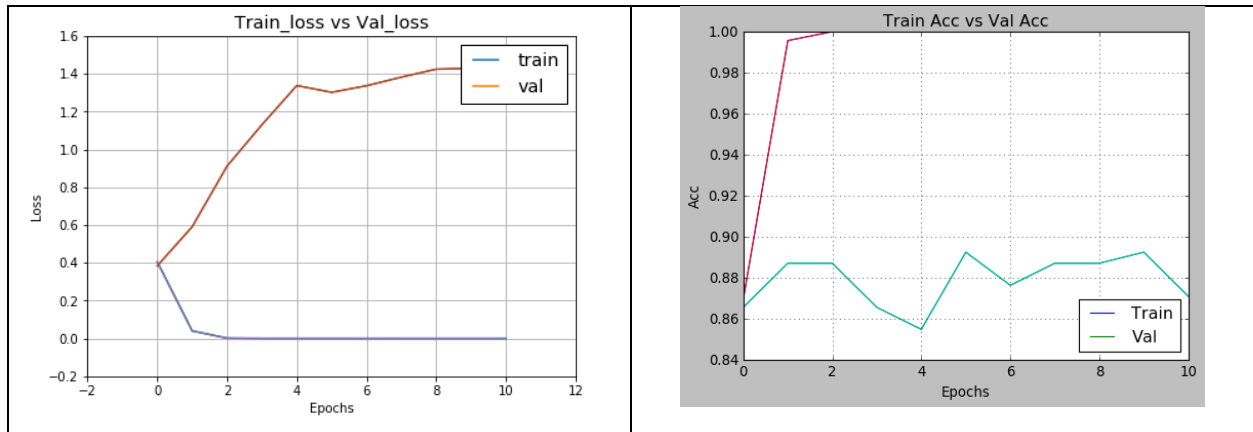


Table 6 - Naive ConvNet Performance during Training

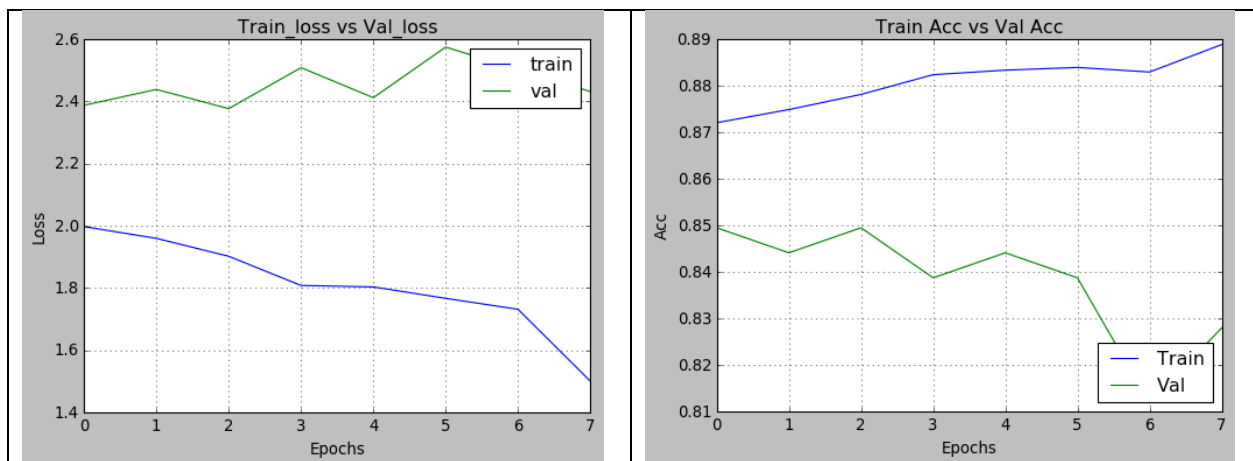


Table 7 - Naive ConvNet Performance during Training using Augmented Data

It is important to stress that the epoch numbers in the diagrams are closely related to the *earlystopping* parameter. The specific parameter is configured with *patience* = 4. Given the visualization in both tables above it is obvious that the models overfit very soon. Many attempts have been made towards optimizing the hyper-parameters, however the values in the submitted notebook, yield the best results.

Accuracy reached almost 83% from the LeNet-Like model, which is not very bad considering the amount of data. However, the models are not robust enough and do not generalize well to unseen data. I strongly believe that the reason behind that is the lack of data. The fact that the model had less than 1000 authentic and fake images is a major issue, which as it seems from the visualizations augmentation did not seem help a lot. Also, I added more extracted frames from FakeApp but the result did not get any better.

Finally, I would trust the LeNet-Like model but I would follow two simple steps in order to put it in production. The first step is to train the models using other metrics than accuracy. Figure 2, describes the imbalance between data, fake faces are less than the authentic, thus I would use f1-score trained in favor of precision since I want the model to be “picky”. Step two, I would ask from users more data in order to feed the model, train it and make it more accurate. However, as it seems from the keras release notes [15] keras does not support other performance metrics than accuracy.

Justification

The results yielded from training/testing are much better than VGG-Face and Match-Face (FaceNet). The proposed architectures were actually able to recognize and classify at least 83% of the times Nicolas Cage authenticity or fakeness compared to VGG-Face which was not able to recognize Nicolas Cage facial structure at all. Also, the proposed model is better than FaceMatch as we observe the results from table 3.

In my opinion, the results from the proposed architectures are a solid to be considered significant enough for further research and development. The performance rates were good enough if someone considering the data provided. I strongly believe that the propose architecture can be the basis in case for further improvement in case more data are available.

VI. Conclusion

Reflection & Improvements

I have proposed and trained an approach for recognizing fake facial structure of people that have fallen victims of misappropriate use of FakeApp. The CNN models are based on custom and LeNet-Like architectures that take as input aligned images of 96x96 in grayscale format. The results although not optimal are good enough to consider the proposed approach as the basis for a solution.

Even though the results are promising, the accuracy rates were not sufficient to put this model in production. One of the next step of this project is to acquire more data from the fake class. Another step, is to optimize the architecture as far as structure and

hyper-parameters are concerned. This step is entirely depended on the data someone has is his possession.

During this project's course I wouldn't say that there were much difficulties apart from fixing the dataset and training time. Extracting and aligning the desired images was long and not so fruitful process. FakeApp extract frames from a video that are either identical or almost the same. That does not serve diversity inside the dataset which is very important the model to be able to generalize on unseen data. Also, I had to convert and resize the data to reduce the training time.

I strongly believe that final model and solution should be used as a benchmark for other researchers. Of course, the implementation needs improvement that is depended on the data but it can definitely be a solution to which other experts can based on.

Literature review did not show any other implementation handling this classification task. However, I do believe that given more data a better solution is possible. Finally, I would like to propose another method based on my research work [16]. During my PhD I proposed a multi-agent system based on reinforcement learning that is cable of exploiting the capabilities of ML algorithms and improve the overall classification performance.

A common problem in ML/DL is that of the imbalanced data. There are lot of effective techniques for tackling this problem [17], however sometimes can be time consuming and tedious task if the data are really messy. It is observed that due to structural differences ML algorithms tend to be better at classifying certain data observations compared to other algorithms. I believe that the same applies to DL, that by changing structure, the parameters and the data distribution, a model can be biased against one class of data. Thus, we can create two or more different model structures (e.g. VGG-16, Xception, AlexNet) and combine their classification capabilities in order to improve classification accuracy. Concluding, between the two models architectures I would prefer the LeNet-Like architecture since it has faster convergence and better performance results.

VII. References

1. *DeepFakes*. Available from: <https://www.highsnobiety.com/p/what-are-deepfakes-ai-porn/>.
2. *DeepFake Porn Industry*.
3. *CNBC Fake Porn Videos*.
4. *PornHub, witter Ban 'DeepFake' AI Modified Porn*.
5. Gunawardana, A. and G. Shani, *A Survey of Accuracy Evaluation Metrics of Recommendation Tasks*. J. Mach. Learn. Res., 2009. **10**: p. 2935-2962.
6. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *ImageNet classification with deep convolutional neural networks*, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. 2012, Curran Associates Inc.: Lake Tahoe, Nevada. p. 1097-1105.
7. Karpathy, A., et al. *Large-Scale Video Classification with Convolutional Neural Networks*. in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014.
8. Papers, D.L. *Deep Learning* Available from: <https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>.
9. *VGG_FACE*.
10. *Keras Blog*.
11. *FaceNet*.
12. *FaceNet GitHub*.
13. Zhang, K., et al., *Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks*. IEEE Signal Processing Letters, 2016. **23**(10): p. 1499-1503.
14. *Kaagle LeNet-like*. Available from: <https://www.kaggle.com/philipxue/leaf-classification-using-cnn>.
15. *Keras Release Notes*. Available from: <https://github.com/keras-team/keras/wiki/Keras-2.0-release-notes>.
16. Kokorakis, V.M., M. Petridis, and S. Kapetanakis, *A Blackboard Based Hybrid Multi-Agent System for Improving Classification Accuracy Using Reinforcement Learning Techniques*, in *Artificial Intelligence XXXIV: 37th SGAI International Conference on Artificial Intelligence, AI 2017, Cambridge, UK, December 12-14, 2017, Proceedings*, M. Bramer and M. Petridis, Editors. 2017, Springer International Publishing: Cham. p. 47-57.
17. *Reasons why your NN is not working*. Available from: <https://blog.slavv.com/37-reasons-why-your-neural-network-is-not-working-4020854bd607>.