Recognition of Animated Sitcom Characters Using Computer Vision and Deep Neural Networks

Sweta Subhra Datta
Department of ECE
North Carolina State University
Raleigh, NC
ssdatta@ncsu.edu

Shreedhar Devendra Kolhe
Department of ECE
North Carolina State University
Raleigh, NC
skolhe@ncsu.edu

Sahil Avinash Deshpande
Department of ECE
North Carolina State University
Raleigh, NC
sadeshp2@ncsu.edu

I. MOTIVATION

Anime and animated sitcoms are the most popular genres in the entertainment industry. Animated sitcoms generated a revenue of 2 million dollars per episode from the year 2015-2020 [6]. With an ever-increasing fan base and ever-growing popularity, it would be substantial to create a model that could detect animated sitcom characters in real-time and give important information about the characters and recommendations to the fans streaming it on online streaming services. The model and the dataset collected are made in such a way so that it remains dynamic and community-driven so that it can be retrained with more data at a later stage [4][5].

Also, a lot of research has been done on the detection of human faces, but there are few studies related to the detection of animated faces [1][8]. Animated faces are very different from human faces because of their tendency to transition very fast in various episodes. E.g., as we can see below, the character of Grandpa Simpson looks different in different episodes. This becomes a considerable challenge during dataset collection and recognition [2].

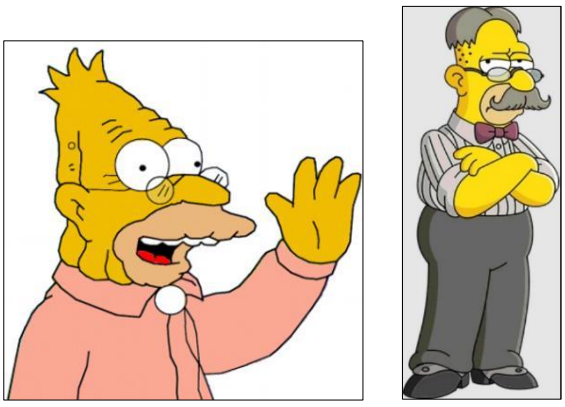

Fig. 1. High Level feature set transition of characters

Although animated sitcom faces have smaller and less global features than human faces, their local features are less defined than the human counterparts. Animated faces, like human faces, also suffer from the problem of occlusion and reduction in feature sets with increasing distance from the source. This project presents an initial approach to identify the popular characters from two animated sitcoms *Simpsons* and *Family Guy*, using Computer Vision with deep neural nets. We also address some of this problem with our dataset augmentation and model training. In contrast, many problems like the problem of occlusion and major feature transition could be solved with the help of more data through the community-driven dynamic dataset[7][8].

II. DATASET DESCRIPTION

As a base, we will be using the *Simpsons Character Data Set*, taken from *Kaggle* for this project. This dataset contains images of 20 characters from the *Simpsons* animated series with 400 images for each of the side characters and 2000 for each of the main characters (*Homer, Marge, Bart, and Lisa*). All images are of size 64x64. For our project we concentrated on detecting 4 classes: *Grandpa Abraham* = Class 0, *Apu* = Class 1, *Bart* = Class 2, and *Charles Montogomery* = Class 3.

In addition to this dataset, we have also created our own dataset consisting of 100 images of *Peter Griffin's* character from the *Family Guy* series. We have made use of an image generator to create this dataset. As we are using supervised learning, there is a label attached for each image class.
This is what the dataset file structure looks like:

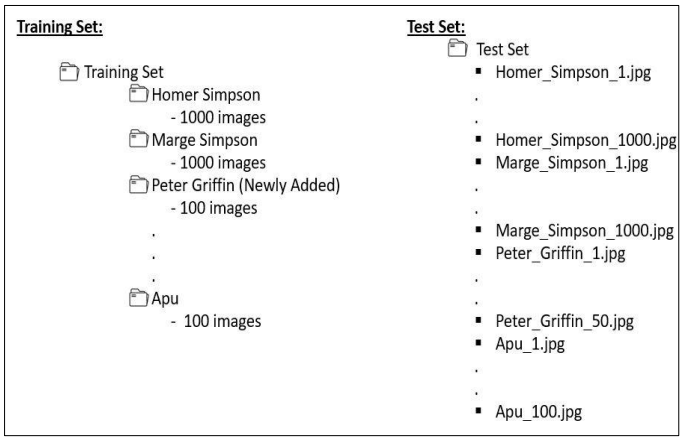

Fig. 2. Dataset File Structure

Apart from these training datasets, we also made a placeholder to make the dataset dynamic and community-driven, where any database collected from different contributors of data of any sitcom characters is added to our databases using a single function/API call. This is similar to how FaceNet [3] has implemented its dynamic dataset. While adding the *Peter-Griffin* data, we just replaced Class 3 (*Charles Montogomery*) with *Peter Griffin*. We could have also added it as an additional class, but we replaced an existing one to maintain uniformity in our test results.
There was no separate validation dataset as we followed convention and used our test data set as our validation set.
We later tested the model on single images in real time. This data was never introduced to the model to validate the performance.

III. METHODOLOGY

A. Data Augmenatation

The data augmentation is done in two phases:

Phase 1 - Weeding out the dirty data first, i.e., the images which did not represent the characters, were removed. This was especially done while we were trying to create our own dataset of the *Peter Griffin* character from *Family Guy.*

Phase 2 - To avoid overfitting, we needed to expand our dataset artificially. The idea is to alter the training data with small transformations to reproduce the variations that may occur during an episode of the series.

When the character is not centered, the scale is not the same, and the character is far away or is poorly represented in a mix of other characters [5]. This transformation is done through a class called *ImageDatagen* from *keras.preprocessing.image*. After passing the required constructor variables like *zoom* = 0.1, rotation range = 0.1, etc., the dataset was randomly increased by two-three times by either zooming in or rotating random images from each dataset. For now, we have not handled occlusion and transition in features problem, instead, we use the trick that if in a video sequence, there are 5 predictions of the character in 5 frames, only then with a confidence level > 90%, we will be showing it to the user. [7]

Comparison	***FaceNet Model***	***Our Model***
Layers	11 Layers CNN + MaxPool + BatchNorm + Dropout	Simplified 5 Layer CNN + MaxPool + BatchNorm + Dropout
Loss	Triplet Loss	Cross Entropy
Embeddings	Yes	No
Face Detection	Uses MTCNN Face Detection to capture training data	MTCNN does not work on animated faces, it needs either handmade data or a more novel architecture
Feature Sets	Local feature sets are well defined and global feature sets are high because of human faces.	Local feature sets are not well defined and global feature sets are low and simple because of animated faces.
Features Transition	Low - either because of lighting conditions or masking (E.g., Glasses)	High
Dataset Implementation	200 Million images used for training the initial weights.	Around 20000 images used for training
Dataset Augmentation	Dataset augmentation not done during initial training, so difficult to recognize faces that are at a distance.	Dataset augmentation was done randomly, where we zoomed in and out randomly on images.
Accuracy	Accuracy of 95.2 % on > 2M test sets	Accuracy of 93.49 % on 200 to 500 test sets of the selected classes
Real Time Recognition	Performs better on human faces, does not work on animated faces.	Designed only for animated faces, but for real time application, accuracy is lower than FaceNet.

Fig. 3. Comparison with FaceNet Model

B. Comparison with baseline

After data augmentation, we started defining our Model architecture. The *FaceNet* model was kept as our baseline [3] model, and a comparison table is given in Fig. 3. The *FaceNet* model was made specifically to detect human faces with well-defined local features and high no global features. The model also involves the use of embeddings, and its data is collected using complex, convoluted architecture *MTCNN*. However, as we are dealing with animated faces, the *FaceNet* architecture gave bad results when it was retrained on our dataset. Firstly, it was challenging to make the model accept our dataset due to dimension issues. Even after resizing the entire dataset and retraining the *FaceNet* model, the accuracy gained on the validation test set was around 40-45% which was unacceptable. We believe this happened because animated faces are synthetic faces. Most of them have a lot of transition in their local features, and the *FaceNet* class embeddings algorithm could not generate embeddings for such faces.

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Fig. 4. FaceNet Architecture

C. Model

As we could not repurpose the FaceNet architecture for our use case, we tried to simplify the FaceNet model architecture rather than repurposing the whole concept. We experimented with two model architectures:

1. **5-layer CNN** and the rest of the layers were kept the same as that of FaceNet, Loss - Cross entropy, Optimizer – Adam, Learning rate - 0.01. Hyperparameters - Learning rate, Batch size, Epochs - 50 and 60 (trial and error). (Final Model)
2. 3-layer CNN, rest of the layers were kept the same as that of FaceNet. Loss - cross-entropy, Optimizer – Adam, Learning rate - 0.01

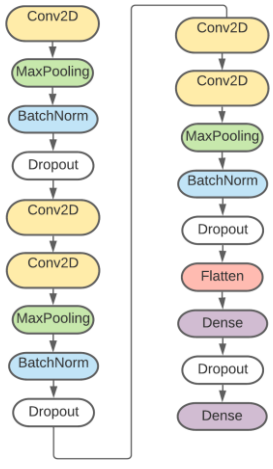

Fig. 5. Final Model Architecture

For optimization of the model, we fed it with images of size 64x64 in *uint* datatype instead of *int64* to save RAM, and like FaceNet, we also used BatchNorm within the layers to speed up the training process.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d (MaxPooling2D)	(None, 32, 32, 32)	0
batch_normalization (BatchNo	(None, 32, 32, 32)	128
dropout (Dropout)	(None, 32, 32, 32)	0
conv2d_1 (Conv2D)	(None, 32, 32, 64)	18496
conv2d_2 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 64)	0
batch_normalization_1 (Batch	(None, 16, 16, 64)	256
dropout_1 (Dropout)	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 16, 16, 86)	49622
conv2d_4 (Conv2D)	(None, 16, 16, 86)	66650
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 86)	0
batch_normalization_2 (Batch	(None, 8, 8, 86)	344
dropout_2 (Dropout)	(None, 8, 8, 86)	0
flatten (Flatten)	(None, 5504)	0
dense (Dense)	(None, 512)	2818560
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 4)	2052

Fig. 6. Model Summary

Hyperparameter	Technique Used
Epochs	GridSearch CV
Batch Size	
Kernel Size	Trial and Error
Dimensions	Random and following convention
Optimizers	Adam( ) was used as it performed well on previous models

Fig. 7. Techniques used for Hyperparameter Tuning.

D. Libraries used:

We have used the *OpenCV* and *NumPy* libraries extensively to read, write, annotate images, image matrix operations, real-time testing on Video sequences, and test individual images. We relied on *Keras* libraries for Data augmentation and the model training and evaluation phase. We also used the *scikitlearn.metrics* library to get the classification report and the *matplotlib.pyplot* to plot the training and validation curves.

IV. Evaluation

A. Testing and Validation plots

As mentioned earlier in section III, there was no separate validation set. Our test set of 200 - 500 images, of the selected classes was made as our validation set. Following are the plots for the models used – CNN with 3 layers and CNN with 5 layers (Final Model)

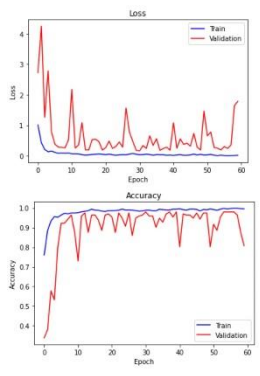

Fig. 8. CNN Model with 3 layers

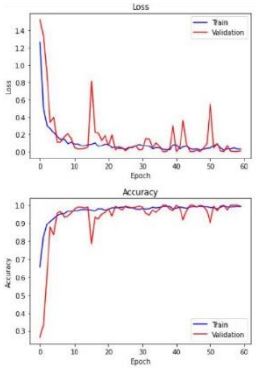

Fig. 9. CNN Model with 5 layers (Final Model)

For the 3-Layer CNN we got a loss of 1.78588 and an accuracy of 80.73% and for the 5-layer CNN (Final Model) we got a loss of **0.00879** and an accuracy of **93.49%**

B. Classification Reports and Confusion Matrix

	precision	recall	f1-score	support
0	0.98	1.00	0.99	48
1	1.00	1.00	1.00	50
2	1.00	1.00	1.00	50
3	1.00	0.99	1.00	44
accuracy			0.99	192
macro avg	0.99	0.99	0.99	192
weighted avg	0.99	0.99	0.99	192

Fig. 10. Classification Metrics for data with only Simpsons characters

	precision	recall	f1-score	support
0	0.98	1.00	0.99	48
1	0.94	1.00	0.97	50
2	0.89	1.00	0.94	50
3	1.00	0.77	0.87	44
accuracy			0.95	192
macro avg	0.95	0.94	0.94	192
weighted avg	0.95	0.95	0.95	192

Fig. 11. Classification Metrics for data with Simpsons characters and Peter Griffin Character (From handmade dataset)

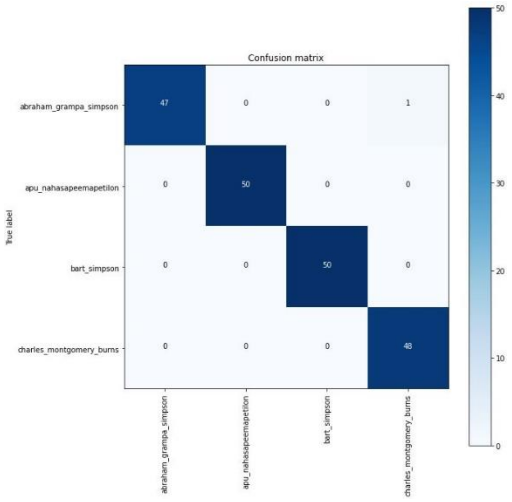

Fig. 12. Confusion Matrix for Simpsons only dataset

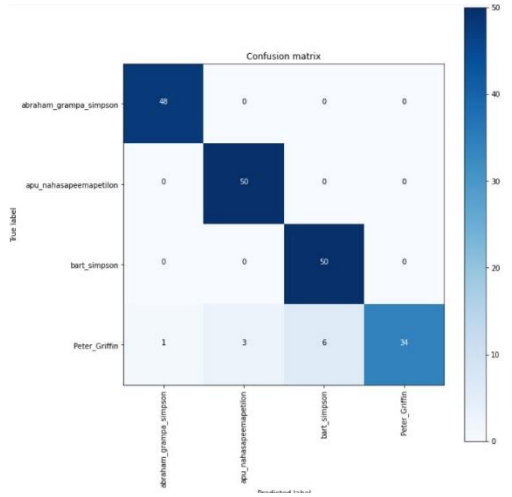

Fig. 13. Confusion Matrix for Simpsons dataset and added Peter Griffin Data

C. Test Case Illustration

The model gave pretty accurate results on the validation set (test set), but on real time images and video sequences, the model did falter a bit in various test cases that we defined which are illustrated below as individual images and tables.

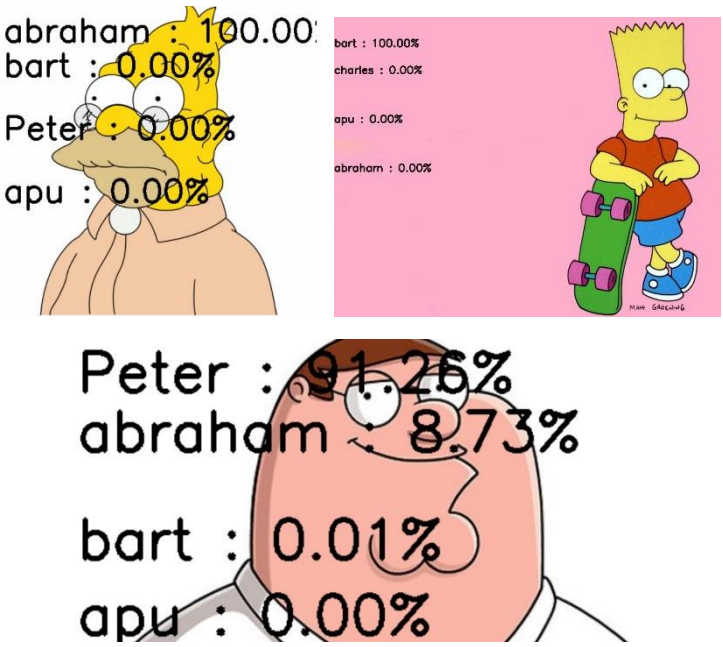

Fig. 14. Test Case Illustration on single images (*Image source: Google Images)*

Test Case	Image	Y_True	Y_Pred	Inference
1	Bart with skateboard	Bart	Bart (Accuracy 100%)	Correct Prediction with very high accuracy
2	Grandpa Simpson	Grandpa Simpson	Grandpa Simpson (Accuracy 100%)	Correct Prediction with very high accuracy
3	Peter Griffin	Peter Griffin	Peter Griffin (Accuracy 82%)	Correct Prediction with very medium accuracy (limited dataset)
4	Apu	Apu	Apu (Accuracy 100%)	Correct Prediction with very high accuracy
5	Human Face	No Prediction	Apu (Accuracy 85%)	Wrong prediction (We need to add human faces as noise in the data for the model to understand difference between human and animated faces.)
6	Homer Simpson with object	No Prediction (Class not added)	Grandpa Simpson (Accuracy 60%)	Wrong Prediction (This can be prevented by training with more Homer Simpson data or increasing threshold of the model)

Fig. 15. 6 Test cases on images (*Image source: Google Images*)

Video content Model Threshold – 85%	Characters in 20 fps	Y_True	Y_Pred	Inference
Simpsons 60 sec. video	Homer Simpson	No Prediction	Apu was detected (False Positive rate of 15/20 fps)	Wrong prediction (Train on Homer Simpson Data)
	Charles, Lisa and Bart	Charles and Bart should be detected	Charles Detected in 10/10 frames and Bart in 5/10 frames	Reasonably good predictions. Display the prediction every 5 to 6 frames to overcome faulty prediction
Human 60 sec. video	Human Faces	No Prediction	Apu and Bart detected in 20/20 frames	Human faces must be added as noise to make the model robust
Simpsons 60 sec. video	Grandpa Simpson, Homer and Bart	Grandpa Simpson and Bart must be detected	In some frames, Homer is detected as bart	Train the model on Homer Simpson Data
	Lisa and Bart Simpson	Bart must be detected	In 7/20 frames Lisa was detected as bart	Introduce Lisa to model as noise or train on Lisa images

Fig. 16. Test Cases on real time video sequence (*Video source: YouTube*)

V. Conclusion

The 5-layer CNN model did well on real time images but, it did falter on a real time video sequence. This can be solved by training all the classes in given the dataset and adding random noise data consisting of human faces and non-sitcom animated characters. In the future, we plan to expand the dataset through contribution from the online community and perform complex data augmentation techniques followed by making the model more similar to the existing FaceNet architecture.

References

[1] Wei-Ta Chu, Wei-Wei Li, "Manga FaceNet: Face Detection in Manga based on Deep Neural Network", National Chung Cheng University Chiayi, Taiwan

[2] Andrew Yip, Pawan Sinha; Role of color in face recognition. *Journal of Vision* 2002;2(7):596.

[3] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

[4] Learning from the Worst: Dynamically Generated Datasets to Improve Online Hate Detection. Bertie Vidgen, Tristan Thrush, Zeerak Waseem, Douwe Kiela

[5] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019)

[6] "Season Program Rankings from 09/22/08 through 05/17/09". ABC Medianet. May 19, 2009. Archived from the original on June 23, 2009. Retrieved July 3, 2009

[7] Cartoon Face Recognition: A Benchmark Dataset

[8] Yi Zheng1, Yifan Zhao2, Mengyuan Ren1, He Yan1, Xiangju Lu1,∗, Junhui Liu1, Jia Li2,∗ 1iQIYI, Inc2State Key Laboratory of Virtual Reality Technology and Systems, SCSE, Beihang University