Comparing and Visualizing GoogLeNet and ResNet on Face Recognition of Animated Characters

Ruoming Ren
University of Toronto - CS Dept.
27 King's College Cir, Toronto, ON
luke.ren@mail.utoronto.ca

Songchen Yuan
University of Toronto - CS Dept.
27 King's College Cir, Toronto, ON
ysc.yuan@mail.utoronto.ca

Abstract

In recent years, we have seen an increased demand on facial recognition for cartoon characters: Disney studio, in 2020, adopted a deep learning model that recognizes specific character information and action sequences in its colossal content archive[17], which assist animators in their production process. In this paper, we will analyze 2 deep learning architectures, namely ResNet-50 and Inception v3, on face recognition of animated characters - we will evaluate the accuracy of outputs of the two models, determine the usefulness of feature extraction[18], visualize and compare filters and feature maps, and finally attempt to interpret and demystify the deep learning processes of the two models with Grad-CAM[7].

1 Introduction

Even prior to the deep learning era, face recognition has been widely used in a myriad of domains from forensic investigation to social media. Researchers around the world have developed various algorithms and models to tackle the challenge with accuracy and efficiency. DeepFace[19], OpenFace[20], and Google FaceNet[21] are popular choices in human face recognition tasks - FaceNet even achieved nearly 100% accuracy in the Labeled Faces in the Wild (LFW) dataset[22]. Notwithstanding this motivating progress, the lack of interpretability of the these complex architectures impedes us from comprehending and validating the decisions made by the models. Utilizing the following tools, we will attempt to analyze and explain the performance of the two models and provide visual explanations to how the models achieve the results.

2 Related Work

GoogLeNet(Inception Net) is a CNN model that is 22 layers deep with some special tools such as 1*1 convolution and Global Average Pooling. Its unique structure is the inception layer. It contains several different sizes convolution layers to process data horizontally before sending the data to the next inception layer. Different sizes of filters inside an inception layer "allows the internal layers to pick and choose which filter size will be relevant to learn the required information"[23] from the image. We will use Inception v3, which is the third version of the GoogLeNet, in this project.

ResNet(Residual Neural Network) is a CNN model with a special structure called Skip Connections. In the modern deep learning era, models get increasingly deeper and more complex[1]. This may result in vanishing and exploding gradients[2] during training. The Skip connections in the model can significantly resolve the problem and render the training process more efficient[3]. This characteristic makes the ResNet be one of the most popular model for image classification. We will use ResNet-50, a ResNet model with 50 convolutional layers.

Preprint. Under review.

Transfer learning is a technique used to apply knowledge gleaned from one domain to another with slight modifications to fit the model to the domain of interest. It is a common approach to take a pretrained CNN model that has been trained on a large dataset, such as ImageNet, and apply it to a relatively smaller dataset (of a similar domain) so that we can take advantage of the feature extraction abilities of the pre-trained model, which can result in high performance. Fine-tuning the pre-trained model can be used to improve the performance.

Grad-CAM(Gradient-weighted Class Activation Mapping) can render the target model more "transparent" by utilizing the gradients to construct a approximate heat map that highlights regions in the input image where the model used to predict the output. Employing Grad-CAM, we are able to understand how the model managed to reach the predictions by looking at the heat maps, from which we can discern what components of the image the model was focusing on and what kinds of features were extracted and learned.

3 Approach

3.1 Dataset Retrieval and Augmentation

Dataset We employed a benchmark cartoon dataset, ICartoon dataset[4], constructed by Zhang, et al. This dataset contains 389678 cropped faces of animated characters(human and non-human) of 5013 identities. By 2020, this is the largest manually labelled dataset for cartoon face recognition according to Zhang, et al. The image labels are mostly correct(with a lower than 5% re-checking error rate), the images are high-quality with high resolution and high variability[4], including different sharpness, angles, and orientation.

Method We randomly selected 80% of the dataset to be the training set and the rest 20% of the data were the validation set. To balance the number of images in training and validation sets for each class, the splitting process was done for images in each individual class. To reduce the training time, we extracted the first 1000 classes out of 5013 classes to feed into our models, resulting in one fifth of the original computation time per epoch.

3.2 Learning Objective

Each image class is labelled by an integer t_i for $i \in \{0, 1, \ldots, 999\}$. We adopted cross entropy loss with softmax activation as the loss function:

$$L(s,t) = -\sum_{i=0}^{999} \log\left(\frac{e^{s_i}}{\sum_{j=1}^{999} e^{s_j}}\right) t_i,$$

where s is the output of the neural network model.

3.3 Training Details

To see if features extracted from human face recognition tasks can be applied to cartoon face recognition, we employed PyTorch's Inception v3 model and ResNet-50 model on pre-trained the ImageNet[6] dataset. The following hyperparamters were used: batch size = 32, number epochs = 10, and learning rate = 0.001. We have also used the Adam optimizer[5] and a PyTorch's learning rate scheduler(with step size = 7 and gamma = 1) to speed up the training process.

To contrast with the performance of pre-trained model, we have also trained the two networks from scratch so that the weights are updated in each epoch.

3.4 Visualization of Feature Maps and Filters

Visualize Filters We obtain the filters of each convolutional layer with the following algorithm[8][9]:

Algorithm 1 Get the filters of the model in each epoch

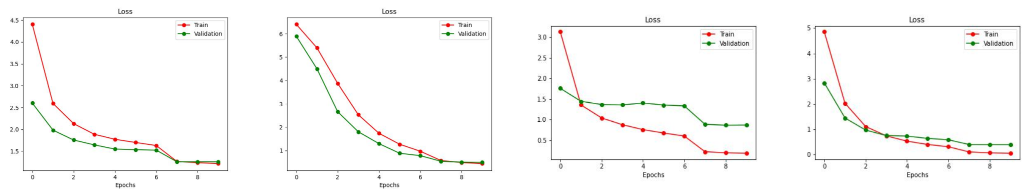
```
numEpochs $\leftarrow$ the number of epochs defined in the training loop(we used 10 in our experiment)
modelWeights $\leftarrow$ a list of size numEpochs, each position is an empty list
for $i = 1$ to numEpochs do
    Train The Model
    numLayers $\leftarrow$ the number of layers in the model
    for $j = 1$ to numLayers do
        append each layers' weights to modelWeights[$i$]
for $i = 1$ to numEpochs do
    for $j = 1$ to numLayers do
        print out modelWeights[$i$][$j$] as graph
```

Visualize Feature Maps After the training and validation processes are over, we save the best model weights and load these weights for feature visualization. Incorporating the use of PyTorch's forward hooks[10], a function registered for the Module object and executed when a forward pass occurs, we are able to construct the outputs of each layer, resulting in the feature maps we would like to obtain.

4 Experiment Results

4.1 Comparing Training Process

Figure 1 exhibits the training and validation losses with respect to the number of epochs. Observe that the initial training and validation losses of GoogLeNet are relatively high compared to those of ResNet - it takes GoogLeNet approximately 6 epochs to reach the same loss as ResNet's in 2 epochs. Also, notice that the performance stops improving after around 7 epochs for all the models, and the training and validation curves stay relatively close to each other and become flat at roughly the same point, indicating that the models are not overfitting or underfitting.



(a) GoogLeNet Pretrained (b) GoogLeNet Not Pretrained (c) ResNet-50 Pretrained (d) ResNet-50 Not Pretrained

Figure 1: Learning Curves for GoogLeNet and ResNet-50

4.2 Comparing Model Performance

Table 1 below illustrates the performance of models with weights updated(not pre-trained) and not updated(pre-trained). Surprisingly, we found that models with pre-trained weights yield a lower accuracy than those trained from scratch. Thus, the features extracted from other objects(including human faces) might not have a great impact on the performance of cartoon face recognition.

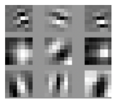
Table 1: Model Performance Comparison by Best Validation Accuracy

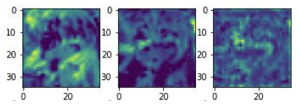
	Inception v3	ResNet-50
Pretrained	73.15%	81.54%
Not Pretrained	87.83%	92.09%

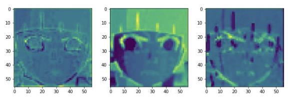
4.3 Comparing Visualizations

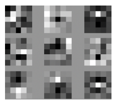
For trained models, ResNet's efficiency of updating weights is much higher than that of GoogLeNet's. This also explains the reason why GoogLeNet needs approximately 6 epochs to reach the loss which only cost ResNet 2 epochs. We can observe from Figure 2 that filters of ResNet are better at edge detection compared with filters of GoogLeNet. This observation is also illustrated by comparing with the feature maps in the next section.

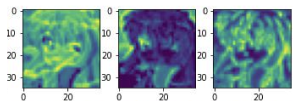
The 2 feature maps in the left column of Figure 2 were produced by the first inception layer of GoogLeNet while the 2 feature maps in the right column were generated by the 10th CNN layer of ResNet-50. It is palpable that models with and without pre-trained weights extract different attributes from the input image; in particular, the pre-trained GoogLeNet model failed to learn any useful features(Figure 2b), whereas the GoogLeNet model without pre-trained weights extract more salient features such as edges of a face(Figure 2e).


(a) ResNet Filter


(b) GoogLeNet Pretrained


(c) ResNet Pretrained


(d) GoogLeNet Filter


(e) GoogLeNet Not Pretrained

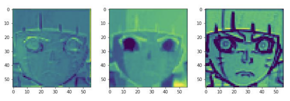
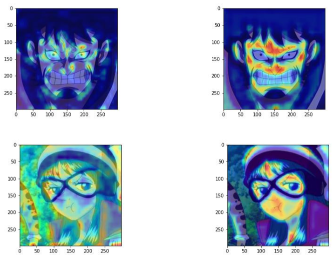

(f) ResNet Not Pretrained

Figure 2: Feature Maps of the 2 Models with/without Pre-trained Weights[10]

4.4 Demystifying Black-Box Models


(a) GoogLeNet

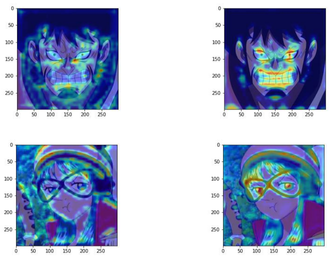

(b) ResNet

Figure 3: Heat Maps Produced by Grad-CAM[11]

For each pair of heat maps, the one on the left hand side corresponds to pre-trained the pre-trained model and the other one corresponds to the model trained from scratch. We can clearly observe that the pre-trained models did not learn the facial information as well as the trained models - the highlighted regions for the pre-trained models do not contain any discernible or salient attributesl, but the highlighted regions for the trained models tell us exactly the features that the model was looking for. Interestingly, the highlighted regions in ResNet's heatmaps seem to be roughly the complement of the highlighted regions in GoogLeNet's heatmaps.

5 Discussion

Here are some aspects on which our research may be extended: use ArcFace[13] and SphereFace[14] as image classifiers and compare their performances; try different models such as DenseNet[15], VGG16[16], etc; compute average drop %[12] and and % increase in confidence[12]; and finally incorporate more feature map visualizations such as gradient visualization with guided backpropagation and saliency maps.

References

[1] Wu, Shen, C., & van den Hengel, A. (2019). Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. Pattern Recognition, 90, 119–133. https://doi.org/10.1016/j.patcog.2019.01.006

[2] Kag, Zhang, Z., & Saligrama, V. (2019). RNNs Evolving on an Equilibrium Manifold: A Panacea for Vanishing and Exploding Gradients?

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

[4] Zheng, Yi & Zhao, Yifan & Ren, Mengyuan & Yan, He & Lu, Xiangju & Liu, Junhui & Li, Jia (2020) Cartoon Face Recognition: A Benchmark Dataset Proceedings of the 28th ACM International Conference on Multimedia, pp. 2264-2272. https://github.com/luxiangju-PersonAI/iCartoonFace

[5] Diederik P. Kingma, Jimmy Ba (2017). Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations. https://doi.org/10.48550/arXiv.1412.6980

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei (2014) ImageNet Large Scale Visual Recognition Challenge. https://doi.org/10.48550/arXiv.1409.0575

[7] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra (2016) Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. https://doi.org/10.48550/arXiv.1610.02391

[8] Rath, S. R.(2020, April 6). Visualizing Filters and Feature Maps in Convolutional Neural Networks using PyTorch. https://debuggercafe.com/visualizing-filters-and-feature-maps-in-convolutional-neural-networks-using-pytorch/

[9] Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Kopf, Andreas and Yang, Edward and DeVito, Zachary and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html

[10]Ayoosh Kathuria (2019) PyTorch 101, Part 5: Understanding Hooks. https://blog.paperspace.com/pytorch-hooks-gradient-clipping-debugging/

[11] Jacob Gildenblat and contributors (2021) PyTorch library for CAM methods. https://github.com/jacobgil/pytorch-grad-cam

[12] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, Vineeth N Balasubramanian (2017) Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks. https://doi.org/10.48550/arXiv.1710.11063

[13] Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou (2018) ArcFace: Additive Angular Margin Loss for Deep Face Recognition. https://doi.org/10.48550/arXiv.1801.07698

[14] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, Le Song (2017) SphereFace: Deep Hypersphere Embedding for Face Recognition. https://doi.org/10.48550/arXiv.1704.08063

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger (2016) Densely Connected Convolutional Networks. https://doi.org/10.48550/arXiv.1608.06993

[16] Karen Simonyan, Andrew Zisserman (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. https://doi.org/10.48550/arXiv.1409.1556

[17] Tarantola, A. (2020, July 17). Disney's new AI is facial recognition for animation. Engadget. Retrieved April 20, 2022, from https://www.engadget.com/disneys-new-ai-is-facial-recognition-for-animation-163054440.html

[18] S. J. Pan and Q. Yang, ”A Survey on Transfer Learning,” in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[19] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1701-1708, doi: 10.1109/CVPR.2014.220.

[20] Brandon Amos, Bartosz Ludwiczuk,† Mahadev Satyanarayanan (2016) OpenFace: A general-purpose face recognition library with mobile applications. https://www.cs.cmu.edu/ satya/docdir/CMU-CS-16-118.pdf

[21] Schroff, F., Kalenichenko, D., Philbin, J. (2015, June 17). FaceNet: A unified embedding for face recognition and clustering. arXiv.org. Retrieved April 20, 2022, from https://arxiv.org/abs/1503.03832

[22] Karam, L. J., Zhu, T. (2015). Quality labeled faces in the wild (QLFW): A database for studying face recognition in real-world environments. Human Vision and Electronic Imaging XX. https://doi.org/10.1117/12.2080393

[23] JalFaizy (2018). Deep Learning in the Trenches: Understanding Inception Network from Scratch. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch

6 Appendix

6.1 Contributions

Ruoming Ren

- Train ResNet and GoogLeNet models
- Visualize the filters and modify the existing algorithm for this task

Songchen Yuan

- Retrieve, load, and split data into training set and test set
- Visualize feature maps and modify the existing algorithm for this task
- Print Heat Maps of images using Grad-CAM