Simpsons Character Recognition Project

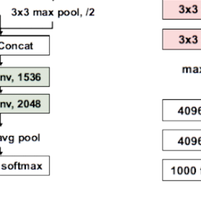
Siqi Li, Lirong Ma, Zhangyi (Rocky) Ye, Haonan Zhang, Yuxuan (Nancy) Zhang, Luping (Rachel) Zhao | MSiA 432 Deep Learning | Spring 2020 | Northwestern University

Problem Statement

The Simpsons is a popular American animated sitcom created by Matt Groening and has been on air since 1989. Due to the large number of characters, sometimes we feel like we have seen a certain character before but don't know exactly who she or he is while watching the show.

This project adopted advanced convolutional neural networks VGG16 / VGG19 (Simonyan & Zisserman 2014) and Xception (Chollet, 2017) to classify images labeled with the 18 selected characters. In addition to image classification, this project further trained Faster R-CNN, an object detection model, to detect and classify images with multiple characters.

The deployment of this project will allow viewers to know which Simpson characters they are watching without pressing pause to check their phones.



Data set

- The Simpsons Characters Data is a public data set created by Alexandre Attia. The raw data contains 20,933 pictures (~45 kB each) for 48 characters.
- As not all characters have that many images, we only kept the top 18 characters each with more than 350 pictures. This gave us a total of 18,992 samples.For these 18 characters, we divided pictures into 3 datasets: Training set (60%), Validation set (20%), an ￥d Test set (20%).
- When training neural network models, we augment the training data by
  - Randomly rotate images with angles 15 and 30 degrees
  - Randomly zoom inside pictures and apply shearing transformations
  - Randomly flip images horizontally and
  - Randomly shift images horizontally/vertically.

Technical Approach

Image Classification

VGG16

- Five convolutional blocks with LeakyReLU activation functions are responsible for feature extraction. Pooling layer takes MaxPooling2D with a pool size of 2x2. The concatenation layer then converts the result from to a vector through the Flatten layer. Dropout layer subsequently excludes a randomly selected percentage of neurons and processes the output from the Flatten layer. Finally, the output layer has neurons and a soft-max activation function to output probability-like predictions for each character.

VGG19

- Similar to VGG16, except it has four convolutional layers in the fourth and fifth block.

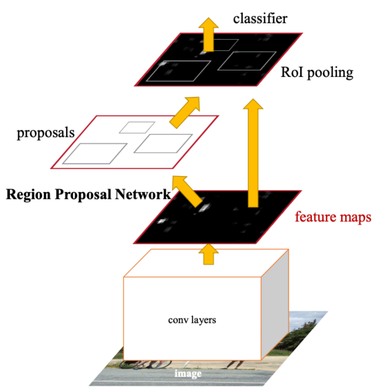
Xception

- Xception, stands for Extreme Inception, merges the idea of GooLeNet and ResNet but replaces the inception modules with a special type of layer, a depthwise separable convolution layer.
- A separable convolutional layer assumes that spatial and cross-channel patterns can be modeled separately.
- It is mainly composed of two parts. The first part applies a single spatial filter for each input feature map, and the second part, a regular convolutional layer with 1x1 filters, only focuses on cross-channel patterns.
- Since separable convolutional layers only have one spatial filter per input channel, it is does not work well for layers that have too few channels, such as the input layer. Therefore, Xception architecture starts with 2 regular convolutional layers and then the rest of the architecture uses only separable convolutions plus a few max pooling layers and the usual final layers, such as a global average pooling layer and a dense output layer.

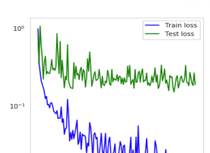
Object Detection

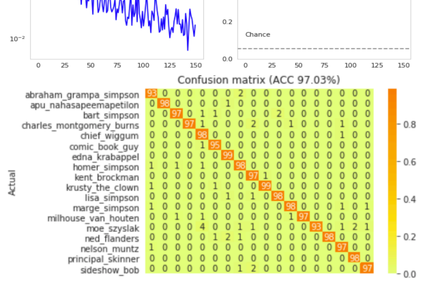
Faster R-CNN

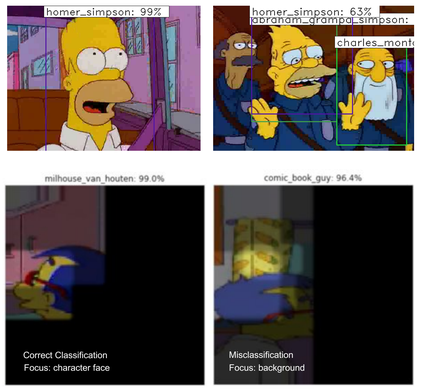
- The Region Proposal Network works by taking the output of a pre-trained deep CNN, such as VGG-16, and outputting multiple region proposals and a class prediction for each.
- Region of interest pooling (RoI pooling) performs max pooling on inputs of nonuniform sizes to obtain fixed-size feature maps.
- The uniformed regions then pass fully connected layers and Softmax layer and make the final classification.

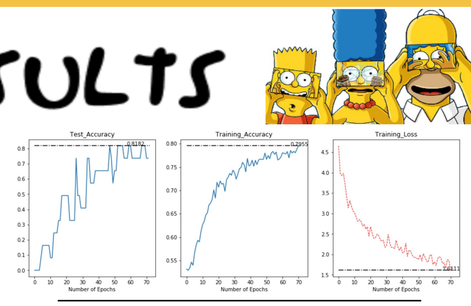


Results









Test Metrics Performance - Top Six Accurate Characters

Character	Precision	Recall	F1 score
ned_flanders	0.913	0.929	0.921
marge_simpson	0.832	0.966	0.894
kent_brockman	0.800	1.000	0.889
principal_skinner	0.870	0.895	0.883
krusty_the_clown	0.826	0.927	0.874
chief_wiggum	0.939	0.795	0.861

- **Best model for classification:** Xception with augmentation. The accuracy achieved 97.03% after 150 epochs. The model performed well in general. Relatively, it performed poorly to recognize Grampa Simpson.
- **Best model for object detection:** the Faster R-CNN model (results shown above) reached a test accuracy of around 73%. Ned Flanders has the highest F1 score, with recall and precision above 90%. This means our model corrected identified 90% of relevant cases among all Ned Flanders cases, and above 90% of the retrieved results are relevant to Ned Flanders. Overall, top six characters achieved the balanced precision and recall above 80%.
- **Heatmaps**: the model focused on character faces and can correctly classified characters most of the time, so we believe the model was reliable. We also noticed when it focused on dissecting the background, the model misclassified the images.
- **Human accuracy**: the model accuracy was close to human accuracy (95%) because of the significant differences among characters in terms of the hair, shape of head, etc. However, the model performed poorly compared with humans, when characters had an unusual hair-style or in different colors than what would be expected. For example, the model failed to recognize characters when they appeared as Lego characters.

Conclusion

- Our Xception model performs close to human with an accuracy level around 97%.
- Model's uncertainty will increase when a character's side face is provided or there are more than one characters in the screenshot.
- We can improve model's performance by including more corner cases in the training set and include labels for every character present in the input picture.
- Our Faster R-CNN model yields and accuracy of around 73% and F1 score of 91%.
- We can improve the accuracy by training further or filling in the missing labels from the data set. We can improve the training and validation speed by using more advanced algorithms like YOLOv4.

References and Related Work

- Attia, A. (2017, June 12). The Simpsons characters recognition and detection using Keras (Part 1). Retrieved from https://medium.com/alex-attia-blog/the-simpsons-character-recognition-using-keras-d8e1796eae36
- Carremans, B (2018, August 17). Classify butterfly images with deep learning in Keras. Retrieved from https://towardsdatascience.com/classify-butterfly-images-with-deep-learning-in-keras-b3101fe0f98
- Girshick, R (2015). Fast R-CNN. Retrieved from https://arxiv.org/abs/1504.08083
- Ren, S et al. (2015) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Retrieved from https://arxiv.org/abs/1506.01497
- Krizhevsky, A et al. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Retrieved from https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- Simonyan, K and Zisserman, A (2015).Very Deep Convolutional Networks for Large-Scale Image recognition. Retrieved from https://arxiv.org/pdf/1409.1556.pdf