Simpsons Character Classification using an Ensemble of DenseNets

Ngoc Dung Huynh
dunghuynh110496@gmail.com
Deakin University
Burwood, VIC, Australia

Figure 1: Simpson Characters

ABSTRACT

Ensemble method is a simple and common technique to improve the accuracy of single classification models. In this work, we ensemble 4 single DenseNet models to build a highly accurate classifier for 20 Simpson characters. Horizontal flipping, zooming, and shearing are used to augment the training data. The result shows that the accuracy on the validation set of the ensemble model is a little higher than the accuracies of individual models (from 99.30% to 99.45%). Additionally, the accuracy on the holdout test set of the ensemble is even higher than at 99.7%, which shows the robustness of the ensemble. The results can be reproduced using the code at https://github.com/parkerhuynh/Simpson-Charater-Detection.

KEYWORDS

Neural Networks, Pre-trained Model, Simpsons Characters, Ensemble Model

ACM Reference Format:
Ngoc Dung Huynh. 2021. Simpsons Character Classification using an Ensemble of DenseNets. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep Neural Network is a machine learning technique inspired by the behaviour of biological neurons in the nervous system [6]. Convolutional Neural Network (CNN) is a class of deep neural networks commonly used in computer vision or analyzing visual imagery [10]. The architecture of a CNN-based image classification network can be divided into 2 major sections: Feature Extraction section and Fully Connected section. The Feature Extraction section consists of a series of convolutional layers and pooling layers. The Fully Connected section consists of a series of fully connected layers. The last fully connected layer has the same number of ouputs as the number of classes we want to predict [7].

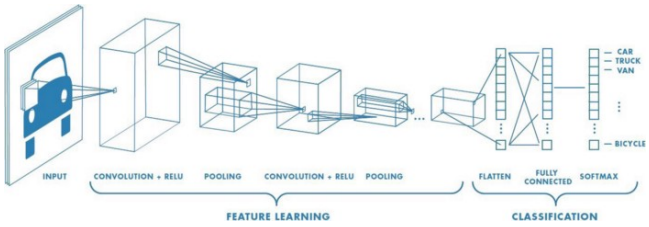

Figure 2: The architecture of CNN.

A pre-trained model is a model already trained using another dataset. The main advantage of a pre-trained model is that we do

not need to train the weights from scratch. As a result, the model is faster to train and it usually leads to a more accurate model due to the effect of transfer learning. The action of optimizing the parameters of a pre-trained model is called fine-tuning. There are some popular available pre-trained models from Keras Application for fine-tuning such as ResNet, DenseNet, and NASNetMobile.

Ensemble techniques combine the predictions by many models to improve the accuracy of individual models. Each model may perform well on certain labels and not the others. Therefore, by combining the predictions from multiple weak models, the models complement each other resulting a higher accuracy overall [5]. On the other hand, combining the predictions from multiple predictors also has a variance reduction effect. As a result, the ensemble is more robust against outliers in the data.

This paper aims to use an ensemble model by combining 4 single pre-trained models (4 DenseNet models) to achieve a high accuracy for classifying the Simpson characters. Firstly, we train four single models (4 DenseNet Models). We then get the predictions on the validation set for each model. Secondly, the final predictions are the argmax of the average probabilities for each class label.

2 DATA

We use the dataset that contains 19,548 labeled images with 20 classes as 20 characters in the Simpson series. We choose a random 80:20 split from the whole dataset for the training and validation sets. The size of each image is 256x256. We use basic data augmentation by using Keras.ImageDataGenerator (rotation_range=30, width_shift_range=0.20, height_shift_range=0.20, brightness_range =[0.6,1.4], zoom_range=[0.6,1.4], shear_range=0.4, and horizontal_flip=True) to address the over-fitting problem.

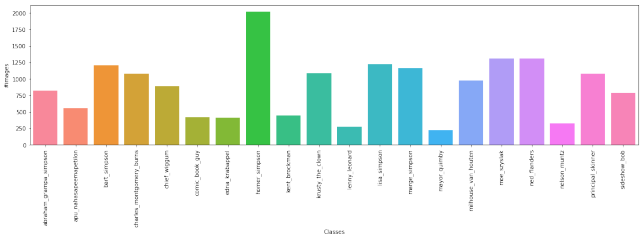

Figure 3: Distribution of images per class.

3 NETWORK DESIGN

We experiment with various network architectures for this problem: ResNet [3], InceptionResNet [8], Xception [2], NasNet [11], EfficientNet [9], and DenseNet [1]. These networks are different variants of convolutional neural networks and they are designed for computer vision tasks such as image classification, object detection, and image segmentation.

ResNet is the first to introduce the idea of skip connections which tremendously ease the process of training very deep neural networks. A common problem which hinders the popularity of very deep network is the gradient vanishing phenomenon, in which the gradients become negligible when they back-propagate to the layers close to the input. The skip connections allow the gradients to flow

Figure 4: 12 example image in the training set.

in the alternate paths avoiding the main path connecting input to output. As a result, ResNet networks are faster to train and can be every deep.

InceptionResNet is the 4th generation of Inception networks proposed by Google Research. It is the marriage between the Inception architecture introduced previously and the skip connection idea introduced in ResNet. The result is that the skip connections accelerate the training of Inception network significantly although the accuracy gap between residual and non-residual Inception networks is small.

Xception was inspired by Inception. In Xception, the Inception modules are replaced by depthwise separable convolutions. As a results, for the similar numbers of parameters, Xception network performs better than Inception network, especially when the dataset is large. This shows that Xception has a more efficient way to utilizing the model parameters than Inception.

Different from all the networks mentioned in this paper, NasNet is not designed by human but by automated architecture search. This technique is very expensive when the dataset is large and should only used for small datasets. The found architecture is then applied to bigger datasets. To prevent overfiting to the particular dataset used for architecture search, the regularization technique called ScheduledDropPath was introduced. At the time when it was introduced, NasNet set various the state of the arts while having less parameters.

EfficientNet studies in depth the model scaling problem via network depth, width, and resolution. As a matter of fact, the performance of a deeper and wider network with higher input resolution is usually better than a smaller counterparts. However, the price is that bigger networks are slower to train and to inference. In addition, arbitrary scaling of depth, width, and resolution may lead to bottleneck in one of them. Therefore, we need to scale all these 3 model sizes concurrently to achieve the best balance between model accuracy and model size.

DenseNet (Densely Connected Convolutional Networks) is another step towards increasing the depth of deep convolutional networks. It's a well-known issue of deep networks that it's harder to train deeper and deeper networks due to the gradient vanishing phenomemon. DenseNets address this issue by skip connections. It is quite similar to ResNet, but it also has a two key differences [1].

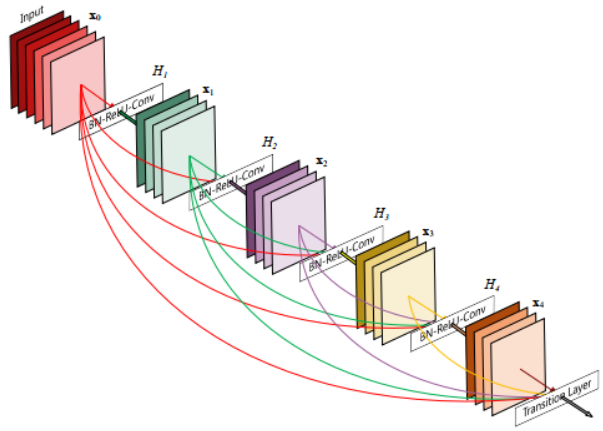

Figure 5: DenseNet with 5 layers.

The first difference is that it has many more skip connections as it connects the output of a layer to all subsequent layers. This further addresses an issue with gradient vanishing commonly encountered in training deep networks. The second difference is the reuse of the feature maps to reduce the number of parameters. An example of DenseNet is illustrated in Figure 5 [1].

To improve the accuracy of a single DenseNet model, we propose an ensemble-based approach to achieve a higher accuracy. We train 4 pre-trained DenseNet models (1 DenseNet121 model, 1 DenseNet169 model, and 2 DenseNet201 models). The structure of these models is presented in Figure 6 [1]. We compute the output probabilities independently of each other. We then take the sum of the probabilities for each class label. After that, the final prediction is the one that has the highest summed probabilities. Our proposed model is presented in Figure 7.

4 TRAINING

All the models are trained using the ADAM algorithm. We train using the batch size of 24 for 64 epochs. The number of GPU is 1 and the environment is Google Colab.

We use the consine learning rate schedule [4] instead of the default schedule provided by the Tensorflow framework. The basic idea of this schedule is warm restarts, which periodically reset the learning rate after a fixed number of iterations. It is hypothesised that this technique helps the network get out of local optima to find better optima. This technique is in contrast to cold restarts, which randomly reset the weights of the network.

5 RESULTS ANALYSIS

According to Table 1, almost all models obtain very high accuracy (higher than 96%+) with the exception of NasNet-Mobile (85.80%). In addition, the performances of the models within the NasNet family is quite disappointing as the biggest model (NasNet-Large) with 88.95M does not perform any better than EfficientNet0 with just 5.33M parameters. The NasNet architecture may have overfitted to the CIFAR-10 dataset used for architecture search and it does not generalize well to this Simpsons dataset.

Among the model families, DenseNet models reach the highest accuracy at 99.36% with DenseNet201. Within the same family of EfficientNet and DenseNet, the numbers of parameters of the models does not matter a lot. The performances of the models within these 2 families are roughly the same.

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)
Convolution	112×112	7×7 conv, stride 2		
Pooling	56×56	3×3 max pool, stride 2		
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv		
	28×28	2×2 average pool, stride 2		
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv		
	14×14	2×2 average pool, stride 2		
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Transition Layer (3)	14×14	1×1 conv		
	7×7	2×2 average pool, stride 2		
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Classification Layer	1×1	7×7 global average pool		
		1000D fully-connected, softmax		

Figure 6: The Architecture of DenseNet121, DenseNet169, DenseNet201 for ImageNet.

Model	Parameters (M)	Validation Accuracy (%)
EfficientNet0	5,330,571	98.85
EfficientNet3	12,320,535	98.47
EfficientNet7	66,658,687	98.97
ResNet50	25,636,712	98.33
Xception	22,910,480	98.66
NasNet-Mobile	5,326,716	85.80
NasNet-Large	88,949,818	97.63
InceptionResNetV2	55,873,736	96.01
DenseNet121	8,062,504	99.29
DenseNet169	14,307,880	99.17
DenseNet201 (1)	20,242,984	99.36
DenseNet201 (2)	20,242,984	99.30

Table 1: Running Time and Accuracy of models

Model	Validation Accuracy (%)	Test Accuracy (%)
Ensemble model	99.45	99.7

Table 2: Accuracy of Ensemble Model

Our best DenseNet model ties with the other two competitors and achieves 99.30% on the holdout validataion set. To further boost the performance of the model, we use the ensembling technique.

We combine our best 4 models, which happen to be DenseNet of different sizes. The accuracy of the ensemble model increases to 99.45%. Comparing to the single models, the accuracy of the ensemble model is a little better. On the holdout test set, the accuracy of the ensemble is even higher at 99.7%, which shows the robustness of the ensemble.

6 CONCLUSION

Ensemble methods have been widely used for improving the results of the single classification model. This method aims to create a strong model by combining multiple weak models. We experiment with an averaging ensemble model by combing 4 single DenseNet models to achieve a very high accuracy for classifying Simpson's characters. The result shows that, compared to the single models, the accuracy of the ensemble model is a little higher with 99.7%.

A limitation of this approach is that each model has an equal contribution to the final prediction made by the ensemble. However, some models perform better than the others. To resolve this issue, in future research, we would like to try the weighted ensemble method that is an extension of the model averaging technique where the contribution of each member to the final prediction is weighted by the performance of the model.

REFERENCES

[1] 2017. Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 4700–4708. https://doi.org/10.1109/CVPR.2017.243

[2] François Chollet. 2016. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR* abs/1610.02357 (2016). arXiv:1610.02357 http://arxiv.org/abs/1610.02357

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[4] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic Gradient Descent with Restarts. *CoRR* abs/1608.03983 (2016). arXiv:1608.03983 http://arxiv.org/abs/1608.03983

[5] Ngoc Tu Pham, Ernest Foo, Suriadi Suriadi, Helen Jeffrey, and Hassan Fareed Lahza. 2018. Improving performance of intrusion detection system using ensemble methods and feature selection. *ACSW 18: Proceedings of the Australasian Computer Science Week Multiconference* 2 (May 2018), 1–6. https://doi.org/10.1145/3167918.3167951

[6] Alex Shenfield, David Day, and Aladdin Ayesh. 2018. Intelligent intrusion detection systems using artificial neural networks. *ICT Express* (May 2018). https://doi.org/10.1016/j.icte.2018.04.003

[7] Indriani Sitorus. 2020. *An Introduction to Convolution Neural Network (CNN) for A Beginner.* Retrieved May 31, 2021 from https://medium.com/easyread/an-introduction-to-convolution-neural-network-cnn-for-a-beginner-88548e4b2a84

[8] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR* abs/1602.07261 (2016). arXiv:1602.07261 http://arxiv.org/abs/1602.07261

[9] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR* abs/1905.11946 (2019). arXiv:1905.11946 http://arxiv.org/abs/1905.11946

[10] Maria V. Valueva, Nikolay Nagornov, Pavel Lyakhov, G.V. Valuev, and N.I. Chervyakov. 2020. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation* 177, 9 (May 2020), 232—-243. https://doi.org/10.1016/j.matcom.2020.04.031

[11] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2017. Learning Transferable Architectures for Scalable Image Recognition. *CoRR* abs/1707.07012 (2017). arXiv:1707.07012 http://arxiv.org/abs/1707.07012