Computer Vision competition to identify different Simpsons characters

Vinayaka Vivekananda Malgi
vmalgi@deakin.edu.au
Deakin University
Melbourne, Victoria, Australia

Figure 1: Deakin Simpsons Challenge Logo, 2021.Photograph by Dr.Mohamed Reda Bouadjenek [Public domain], via Github. (https://bit.ly/3pncTpa)

ABSTRACT

Deakin Simpsons challenge is a computer vision competition conducted by Deakin university whose goal is to recognize the individual characters of Simpsons family.The characters in the images are recognized by using Computer vision algorithms in Machine Learning. A total of 19,548 labeled images were provided to train the model and tune the hyper-parameters. To identify the characters of Simpsons family we utilized a widely used technique in the field of Deep Learning called transfer learning . The results indicated that the pre-trained model was able to identify about 98.2 % of Simpsons characters on the test data. Finally, we discuss these results to understand the model's performance and accuracy.

1 INTRODUCTION

The performance of the computer vision models has been improved significantly by deep neural networks that take advantage of large quantities of labeled data.But, the models developed and trained on one data set might perform poorly on another but slightly related data sets.This shortcoming calls for the adaptation of strategies that allows the model to perform better on other related data sets.For this we train the model on a already existing data set and then use the same trained model to perform tasks on other related data sets.This approach is known as Transfer Learning and is widely used in field of Deep Learning [2].

In this report, we will discuss on how Computer Vision can be utilized on classifying the various Simpsons characters.Simpsons family is an American animated sitcom distributed by Fox broadcasting corporation LLC,USA [3].Additionally, this report also discusses on how we effectively utilized an already pre-trained model called 'ResNet152V2' to identify the various Simpson's characters.This model was chosen for identifying the various simpsons characters as it's a state of the art model and can be easily generalized over other images provided in this competition

Although, there is a significant amount of work that has been done on this competition,I believe that further improvements can still be made and achieve much higher benchmarks by using more recent advance deep learning models like NasNet and Efficient-Net.In all settings, the Resnet model that we used was able to identify most of the Simpson's characters accurately.

2 ABOUT DATASET

The dataset that was provided in this task contained 19,548 labeled images of different Simpsons characters.These labeled images were used to train our model and then tune the hyper parameters.Additionally, in order for the model to perform better, we extended the dataset by using data augmentation techniques.

3 SETTING UP THE ENVIRONMENT

Before training the model, I first set up the environment in Google Colab. I used Google Colab as our environment for training the model as I used an M1 version of Mac and it needed an entirely different version of tensorflow to train the model that wouldn't be compatible with tensorflow version of the CodaLab Environment where we validate the model on private test data. The below table summarizes the environment settings used in Google Colab for training the model :

Table 1: Google Colab Environment settings

Environment Settings
Tensorflow version - 2.2.0
Python version - 3.7
GPU - NVIDIA Tesla P100
GPU RAM - 16GB
CPU RAM - 32GB

After setting up the environment needed to develop and train the model we downloaded the dataset from the link that was already provided in the example notebook.

4 PRE-PROCESSING THE IMAGE DATA

After setting up the environment and downloading the image data set, the next step was to pre-process the image data set so that it could be used for training the model. Since, we decided to use Resnet152v2 for training our model, it was very necessary for the image size to be set to 224 x 224 as Resnet152v2 was trained on images having image dimension of 224 x 224[1].We set the batch size to be 64 and we created an image pipeline that would handle the image pre-processing for both train and test data.For train data, we randomly rotated images by 20 degrees,randomly created a zoom range of 10 %.Additionally, we also randomly flip the images so that model wouldn't be accustomed to those images.Furthermore, we randomly shifted the images both horizontally and vertically by a factor of 0.2.We then split the data into train and test in the ratio of 0.8 and 0.2.Moreover, we also reserved a few images for validation purposes.

By rotating the images randomly and shifting the height and width of the images we were able to achieve data augmentation of images which resulted in extra addition of images along with already existing images.These new additional images will also be used in training the model, thereby resulting in better fitting of the model.

Next,we created data pipeline called train_generator and test_generator, that would act as pipelines of the model for both train and test data.

5 DESIGNING THE MODEL

Initially, when I first started to design the model for recognizing the Simpson's characters it was very basic model with ten convolutional layers and four layers of deep neural networks.The model was not over-fitting on the validation dataset and the results of the model were not very promising.Even we tried to tune the hyper-parameters by using keras tuner, which resulted in a very small improvement, yet the results were still not promising.This made us to think, if we can apply any other technique that would result in better identification of the Simpsons character.Thus we explored whether Transfer Learning could be applied to classify the Simpsons characters.

Transfer learning as we know is a widely used technique in the field of Deep-Learning that allows us to use the pre-trained models for performing any kind of task say for example image classification, object detection.These models would already be pre-trained on an exisiting dataset say like COCO dataset,Imagenet dataset and we could easily use the same model for classifying similar related images.This would save time and help us to achieve the task much faster as we know building and training a deep Convolution Neural Network is extremely time consuming task.

Choosing a pre-trained model was little challenging for us as there are lot of pre-trained models available in the keras applications and after researching a little bit we chose VGG19,Resnet152V2, InceptionResnetV2 and EfficientNet.Finally, we narrowed down to Resnet152V2 as it had the best performance among the shortlisted models.We omitted InceptionResnet and EfficientNet as they needed a higher version of tensorflow and CodaLab had tensorflow version 2.2.0.

In next subsection, we will discuss the architecture of Resnet152v2 model in detail.

5.1 Architecture of the model

The architecture of Resnet152V2 is similar to Resnet 50 except it has 152 layers of convolution blocks that are stacked upon each other when compared to resnet 50 which has 50 layers only[1].On top of these 152 layers we have added a flatten layer and then we have connected with a dense neural network consisting of four layers.Each of these layers consists of 1024,1024,512,20 neurons respectively.The final layer has 20 neurons only as we have to identify 20 different Simpsons characters.As a result of this, this architecture has total of 156 trainable layers which translates to approximately 6.2 million total parameters.Out of this around 6.1 million parameters are trainable whereas the rest of the parameters are simply not trainable.

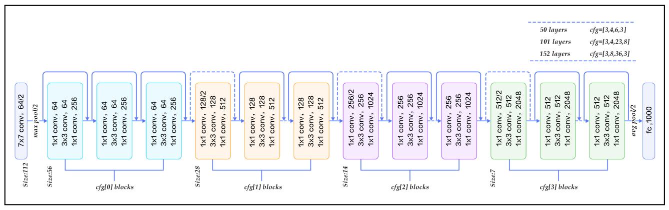

50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]
7x7 conv, 64/2
max pool/2
1x1 conv, 64 3x3 conv, 64 1x1 conv, 256
1x1 conv, 128/2 3x3 conv, 128 1x1 conv, 512
1x1 conv, 128 3x3 conv, 128 1x1 conv, 512
1x1 conv, 256/2 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 256 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 512/2 3x3 conv, 512 1x1 conv, 2048
1x1 conv, 512 3x3 conv, 512 1x1 conv, 2048
avg pool/2
fc, 1000
Size:112
Size:56
Size:28
Size:14
Size:7
cfg[0] blocks
cfg[1] blocks
cfg[2] blocks
cfg[3] blocks

Figure 2: Architecture of Resnet151v2 model.Photograph by Analytics Vidhya [Public domain], via Medium Inc. (https://bit.ly/3g8vmlb)

5.2 Loss Function

The model uses *categorical cross entropy* as it's loss function as there are 20 different classes.Furthermore, we felt that categorical cross entropy is the best loss function that can be applied in our case as we have to classify 20 different Simpsons characters.

5.3 Optimizer

The model uses *Adam optimize*r as Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. Adam is relatively easy to configure where the default configuration parameters do well on most problems.Also, in order to speed up the convergence of the model we have utilized the Adam optimizer.

5.4 Metrics

We have used *accuracy* as the performance metric as we know that this is a multi-class classification problem.Therefore, the best metric for multi-class classification problem is the accuracy.

5.5 Learning Rate

We have implemented a dynamic Learning Rate that basically ramps up during the initial training epochs.In other words, to speed up the model training initially we increase the learning rate exponentially and then decrease the learning rate exponentially.We have set the learning rate to decrease exponentially at 0.8.The learning rate starts at 0.0001 and ends at 0.0005 with learning rate being ramped up at every 5 epochs.This results in better training of the model and also prevents the overfitting of the model.Care has been taken to ensure that learning rate has been set at minimum because if we start with a high learning rate then chances are that we may break the existing fine tuned weights

5.6 Callbacks

The model has been implemented with callbacks to ensure that best model weights are stored periodically.Also we have used model checkpoints that monitors the validation accuracy and saves the best weights of highest accuracy model.Additionally, we have also implemented early stopping to stop training the model in case the model accuracy doesn't increase for next 10 epochs. Moreover, in order to prevent the learning rate from plateauing we have implemented *ReduceLRonPlateau* so that learning doesn't stagnate as the training progresses.All these callbacks along with Learning Rate have been stored in a list called *callbacks_list* and this list will be called during training process.

6 TRAINING THE MODEL

This model was trained for total of 40 epochs starting with a learning rate of 0.0001.Since, the image size is huge we cannot train all the images at once.Hence we divided the images into batches of 245 each and these batches are passed into the model.One epoch is when the entire dataset is passed forward and backward through the neural network.Since the batch size is very small, given the dataset size, the model takes lot of time to train to complete all the 40 epochs.Additionally, we called the callbacks here to improvise the training process.Furthermore, we noticed that as the training progressed both the training accuracy and the validation accuracy increased.

7 PERFORMANCE OF THE MODEL

7.1 Training and Validation loss

From the above graph we can infer that as the training progressed there was a dramatic drop in loss on both the training side as well as the validation side.However, we can notice that around epoch

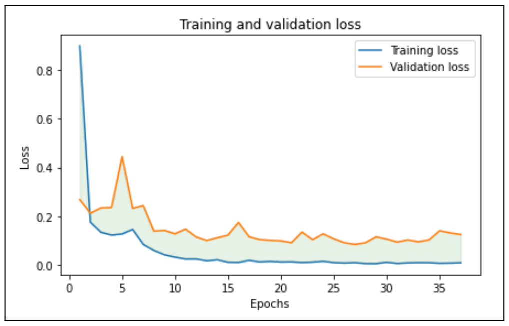


Figure 3: Training and Validation loss

number 3 the training loss is much lesser than the validation loss. This confirms that model has overfitted a little as we all know that Deep learning models are over-parameterized models as the parameters of the model exceeded the size of the training dataset.Due to their over-parameterized nature, these models often have the tendency to over-fit.

7.2 Training and validation accuracy

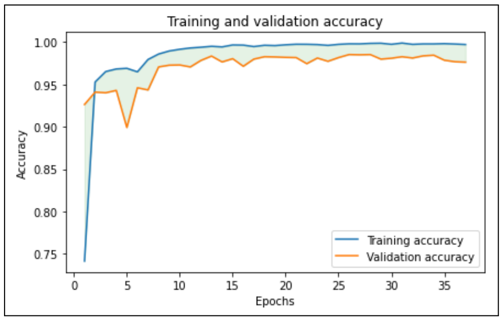
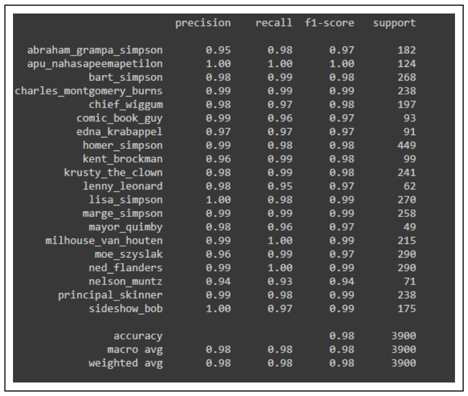


Figure 4: Training and Validation accuracy

From the above graph we can infer that as the epochs increased there was a gradual increase in accuracy on both the training side as well as the validation side.When the model was training around third epoch we noticed that train accuracy was much higher than validation accuracy. This confirms that model was experiencing a little bit of over-fitting.

7.3 Classification Report

From the above classification report, we can conclude that accuracy of the model is around 98%.This means that the model is able to identify 98% of Simpsons characters accurately.Also, if we look at the support column, we can see that images are not balanced as per the classes.The precision of the model is also excellent as we can see that it was able to identify all images of *sideshow_bob,lisa_simpson and apu_nahasapeemapetilon* accurately.Furthermore, even the recall and the F1 scores of the model are excellent.



```
                          precision    recall  f1-score   support

   abraham_grampa_simpson       0.95      0.98      0.97       182
   apu_nahasapeemapetilon       1.00      1.00      1.00       124
             bart_simpson       0.98      0.99      0.98       268
charles_montgomery_burns       0.99      0.99      0.99       238
             chief_wiggum       0.98      0.97      0.98       197
            comic_book_guy       0.99      0.96      0.97        93
            edna_krabappel       0.97      0.97      0.97        91
             homer_simpson       0.99      0.98      0.98       449
             kent_brockman       0.96      0.99      0.98        99
          krusty_the_clown       0.98      0.99      0.98       241
             lenny_leonard       0.98      0.95      0.97        62
              lisa_simpson       1.00      0.98      0.99       270
             marge_simpson       0.99      0.99      0.99       258
              mayor_quimby       0.98      0.96      0.97        49
       milhouse_van_houten       0.99      1.00      0.99       215
               moe_szyslak       0.96      0.99      0.97       290
              ned_flanders       0.99      1.00      0.99       290
              nelson_muntz       0.94      0.93      0.94        71
         principal_skinner       0.99      0.98      0.99       238
              sideshow_bob       1.00      0.97      0.99       175

                  accuracy                           0.98      3900
                 macro avg       0.98      0.98      0.98      3900
              weighted avg       0.98      0.98      0.98      3900
```

Figure 5: Classification Report of the model

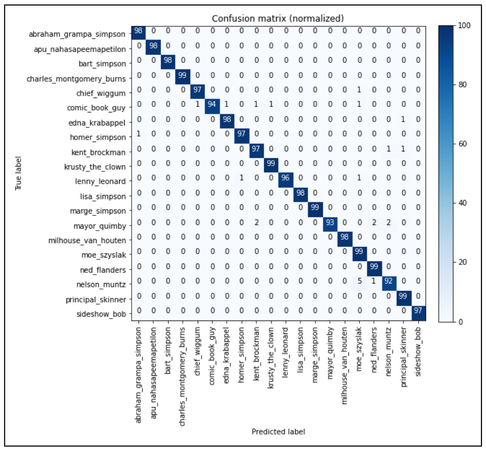


Figure 6: Confusion matrix of the model

7.4 Confusion Matrix

Another important performance measurement for machine learning classification when the output is 2 or more than 2 classes.From the confusion matrix,we can say that model is well able to classify most of the characters accurately and most of the percentages are in nineties and above.The only character which the model struggled to classify is the character *nelson_muntz* where it was able to classify only 92% of such images correctly.

8 GITHUB LINK

The code for this solution has been uploaded to my Github repository in the below link.The code is free to use for educational purposes only. **Github link** : My submission

9 ANALYSIS OF THE PERFORMANCE OF THE RESNET152V2 MODEL

From the subplot of the images,we can see that the model was able to identify most of the characters accurately from the validation data set.We also see that it was able to identify the images even when the images were rotated slightly and it's height and width were changed. The performance exhibited by this model is

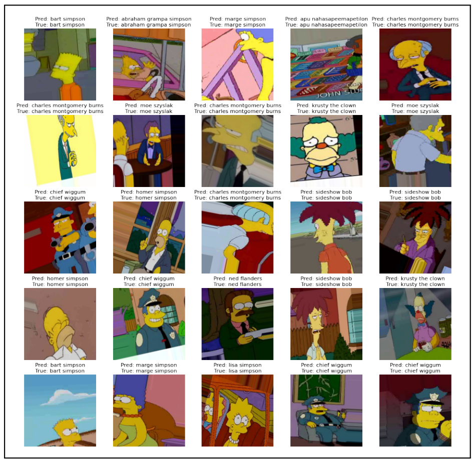


Figure 7: Analysing the performance of model classification

excellent in stark contrast to the basic model whose performance was exceptionally bad.This goes on to show that Transfer learning is indeed one of the best techniques to apply for image classification.

10 CONCLUSION

The Resnet152v2 model was able to achieve an accuracy of 98.2% on the final test data in the CodaLab Environment.By applying transfer learning technique and training for around 40 epochs our model was able to achieve state of the art results in identifying the Simpsons characters.The Resnet152v2 is a very deep neural network architecture that is already trained on a hard and large dataset and achieves an accuracy of 98.2% in just 40 epochs for Simpsons dataset.The success of this architecture will make identification of other class of images very easier.

11 ACKNOWLEDGMENTS

I would like to give my acknowledgments to School of Information technology, Deakin University for giving me this wonderful opportunity to compete in a computer vision challenge. Furthermore, I would like to extend my sincere thanks to Dr.Mohamed Reda Bouadjenek whose valuable support helped me to resolve the environment issues in Codalab and also to the committee for taking time out of their busy schedule to evaluate our submitted codes and reports despite the extraordinary times we are facing in.

REFERENCES

[1] Keras team. 2021. ResNet and ResNetV2. https://keras.io/api/applications/resnet/#h3. [Online; accessed 5-June-2021].

[2] Jing Wang, Jiahong Chen, Jianzhe Lin, Leonid Sigal, and Clarence W de Silva. 2021. Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by Gaussian-guided latent alignment. *Pattern Recognition* 116 (2021), 107943.

[3] Wikipedia contributors. 2021. The Simpsons — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=The_Simpsons&oldid=1025780270 [Online; accessed 5-June-2021].