DEAKIN SIMPSONS CHALLENGE 2022

Building a machine learning/deep learning model in answering Yes/No questions using Simpson images

Gavinn Fernando
Deakin University
Burwood VIC, Australia

Figure 1. Simpson Characters

ABSTRACT

Artificial intelligence with Machine Learning (ML) and Deep Learning (DL) are increasingly helping humans in all levels of activities they engage in. Convolutional Neural Networks (CNN) are mostly used in image classifications networks where DL is used. Recurrent Neural Network (RNN) Architecture and CNN are jointly used in this experiment. With a trained model to achieve high accuracy results with development data, it is important to spend more time in training data to fix the "overfitting" issue as it is one of the key factors to achieve high accuracy results in development model.

KEYWORDS

Neural Networks, overfitting, accuracy

1. INTRODUCTION

In a technologically developing world Machine Learning and Deep Learning are very popular subjects and methods used in almost every industry to predict, analyze and provide near accurate results that are expected without human intervention. For example, Microsoft is using these technologies and investing heavily on ML to catch advanced cyber-attacks when implementing Windows Advanced Threat Protection (ATP) to office 365 domain of companies (Microsoft 2017). This paper highlights the problem to be solved and the methods used to reach the highest possible level of accuracy to solve it by using Tensorflow framework in "Google Colab" where the training takes place to create a model to be used in development environment with the development data, that is not downloadable.

In this paper my focus was on binary (yes/no) solutions rather than open ended questions similar to (Q: "What is the man doing ") as per the below picture figure. 2, where all the relevant words in the vocabulary used including "sitting" exists in the question alone. One interesting observation to note in this picture below is , without a machine learning or deep learning technique is applied to the image, through our naked eye, the answer is clear due to the resolution of the image that is very high.

Figure 2. A clear image of a man "sitting"

Similarly, if we look at the below very low-resolution images (Figure 3.) , it is not possible to easily answer even an binary question with "yes/no" solution by looking at them. For example, the first image with a question "is there a bed in the picture "?. But the ML and DL techniques available could be used to predict the answer for such questions with very low-resolution images as below very much accurately where all the relevant words such as "bed" available in the vocabulary which is quite amazing.

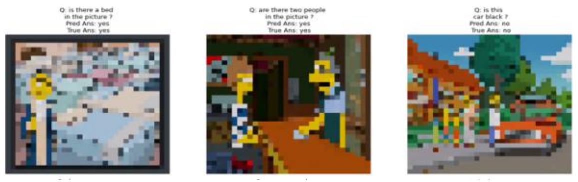


Figure 3. Low resolution images for binary (yes/no) answers

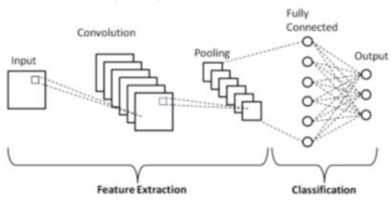
In order to meet this complex challenge, the method below was used to achieve the best results possible in this exercise.

2. METHOD

In image classifications, Deep Neural Networks (DNN) are used very effectively due to the advancements in Convolutional Neural Networks (CNNs) (Shorten and Khoshgoftaar 2019). These networks are able to employ parameterized lightly merged kernels by preserving the special characteristics of the images that are on test.

CNNs are the mostly used networks in image classifications when DL is used. They are used in image classifications and many other state-of- the-art detections and recognitions. Feature extractions and classifications are combined in CNNs.

Figure 4. Schematic Diagram of a Basic convolutional neural network
(CNN) architecture



Source: (Phung VH and Rhee EJ 2019)

3. DESIGN

Recurrent Neural Network (RNN) Architecture combined with CNN are used in this experiment. Similar to CNN, RNN utilizes training data to learn. RNN is another type of neural network that utilizes time series data or sequential data. Traditional deep neural networks assume that the inputs and the outputs are mutually exclusive whereas RNN outputs are depended on prior elements within the sequence (IBM 2020). These DL algorithms are generally used for image captioning, natural language processing etc. Image captioning is a textual description of an image that could be labeled as in Figure 3 above the images in this exercise. While CNN encodes the image captured and the RNN does the language modelling by decoding the encoded output given by CNN (Radhakrishnan 2017).

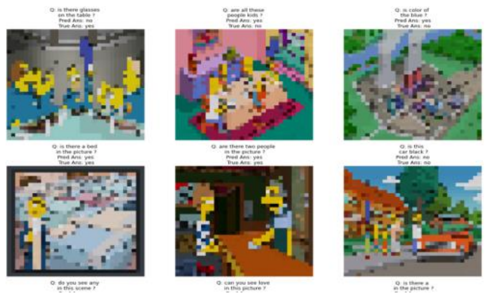


Figure 5. Three images per row

4. IMPROVING ACCURACY OF IMAGE RECOGNITON MODELS

The whole exercise of this challenge was to improve the accuracy of the training dataset and create a model, that is to be used in development stage. To improve the accuracy of the model, I used the techniques below.

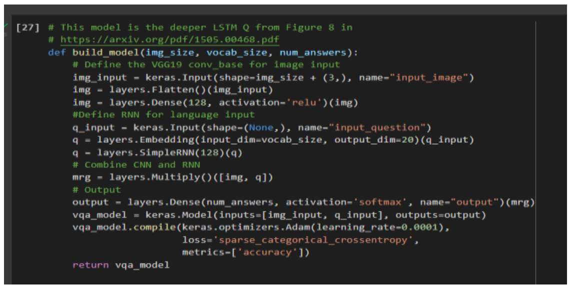
4.1 GETTING MORE DATA

Instead of 3 images data set in a row I increased it to 6 images per row x 5 rows.

Figure 6. 30 images

4.2 ADDING MORE LAYERS AND CHANGING LEARNING RATE

For the testing model, added more layers in order to increase its ability to learn the data more deeply. Dense layers and RNN layers increased to 128 while 3 epochs and three images set per row and the image sizes 32 x 32 were in place. Also changed the learning rate from 0.001 to 0.0001.



```
[27] # This model is the deeper LSTM Q from Figure 8 in
     # https://arxiv.org/pdf/1505.00468.pdf
     def build_model(img_size, vocab_size, num_answers):
         # Define the VGG19 conv_base for image input
         img_input = keras.Input(shape=img_size + (3,), name="input_image")
         img = layers.Flatten()(img_input)
         img = layers.Dense(128, activation='relu')(img)
         #Define RNN for language input
         q_input = keras.Input(shape=(None,), name="input_question")
         q = layers.Embedding(input_dim=vocab_size, output_dim=20)(q_input)
         q = layers.SimpleRNN(128)(q)
         # Combine CNN and RNN
         mrg = layers.Multiply()([img, q])
         # Output
         output = layers.Dense(num_answers, activation='softmax', name="output")(mrg)
         vqa_model = keras.Model(inputs=[img_input, q_input], outputs=output)
         vqa_model.compile(keras.optimizers.Adam(learning_rate=0.0001),
                           loss='sparse_categorical_crossentropy',
                           metrics=['accuracy'])
         return vqa_model
```

Figure7. Dense 128 RNN 128

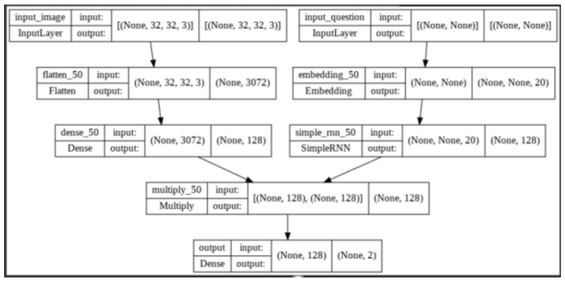
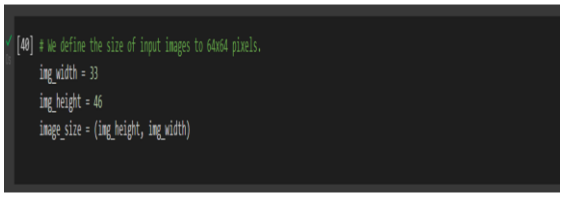

input_image | InputLayer | input: | output: | [(None, 32, 32, 3)] | [(None, 32, 32, 3)]
input_question | InputLayer | input: | output: | [(None, None)] | [(None, None)]
flatten_50 | Flatten | input: | output: | (None, 32, 32, 3) | (None, 3072)
embedding_50 | Embedding | input: | output: | (None, None) | (None, None, 20)
dense_50 | Dense | input: | output: | (None, 3072) | (None, 128)
simple_rnn_50 | SimpleRNN | input: | output: | (None, None, 20) | (None, 128)
multiply_50 | Multiply | input: | output: | [(None, 128), (None, 128)] | (None, 128)
output | Dense | input: | output: | (None, 128) | (None, 2)

Figure 8. 32 x 32, 3 images per row

4.3 CHANGING IMAGE SIZES

I increased the height of the images from 32 to 46 and the width 32 to 33. This was done to enable capturing of the distinctive features of the images, if not covered by the original image size.

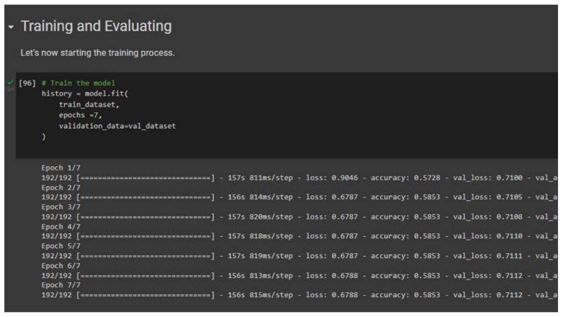


```
[40] # We define the size of input images to 64x64 pixels.
     img_width = 33
     img_height = 46
     image_size = (img_height, img_width)
```

Figure 9. 46 x 33, 3 images per row

4.4 INCREASING EPOCHS

Epochs define the number of times the training passes the neural network. Since the training data is not that large, I had to keep that to a lower value since increasing epochs would not improve accuracy after it reaches a specific point.



Training and Evaluating

Let's now starting the training process.

```
[96] # Train the model
     history = model.fit(
         train_dataset,
         epochs =7,
         validation_data=val_dataset
     )

     Epoch 1/7
     192/192 [==============================] - 157s 811ms/step - loss: 0.9046 - accuracy: 0.5728 - val_loss: 0.7100 - val_a
     Epoch 2/7
     192/192 [==============================] - 156s 814ms/step - loss: 0.6787 - accuracy: 0.5853 - val_loss: 0.7105 - val_a
     Epoch 3/7
     192/192 [==============================] - 157s 820ms/step - loss: 0.6787 - accuracy: 0.5853 - val_loss: 0.7108 - val_a
     Epoch 4/7
     192/192 [==============================] - 157s 818ms/step - loss: 0.6787 - accuracy: 0.5853 - val_loss: 0.7110 - val_a
     Epoch 5/7
     192/192 [==============================] - 157s 819ms/step - loss: 0.6787 - accuracy: 0.5853 - val_loss: 0.7111 - val_a
     Epoch 6/7
     192/192 [==============================] - 156s 813ms/step - loss: 0.6788 - accuracy: 0.5853 - val_loss: 0.7112 - val_a
     Epoch 7/7
     192/192 [==============================] - 156s 815ms/step - loss: 0.6788 - accuracy: 0.5853 - val_loss: 0.7112 - val_a
```

Figure 10. increased epochs from 3 to 7.

5. RESULTS

With all the above tests done to improve the accuracy, the final model of the 4.2 was used to apply to the development data. Adding more layers, keeping the image size as it is with 4 epochs and changing the learning rate from 0.001 to 0.0001, the following result achieved.

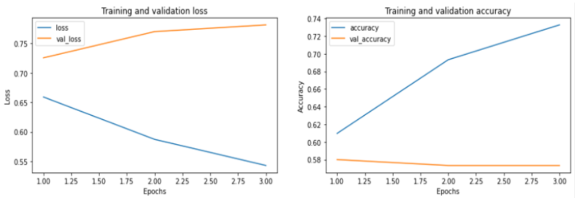


Figure 11. Training and validation

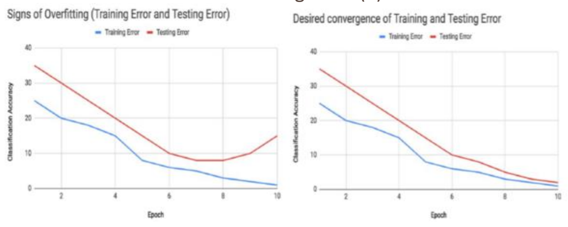
This result indicates that the model generated with test data would not provide high accuracy as per 4.2 above. The reason for that was, in this model, when the training rate continues to decrease, the validation loss increases, resulting in “overfitting “condition which impedes the higher accuracy in results. Accordingly, when applying this model to the development set, ended up obtaining a result of 50.4 accuracy.

6. CONCLUSION

To conclude this paper, in summary I have explained the building process of a Deep Learning model with technical briefing of CNN and RNN and how they interact in digital model designing. Also, I have explained the factors to be considered in increasing the accuracy of the model regardless of the outcome of the results of the accuracy achieved in development test data set. It is noticeable that spending more effort in training data to fix the “overfitting” issue as it is identified is an important factor. In order to achieve very high accuracy with the development dataset, a good model will enhance a high success rate.

Below is a good example of overfitting issue and a comparison of such situation with desired relationship with testing and training errors.

Figure 12. Overfitting plot vs desired relationship of training and testing error (6)



Source: (Shorten and Khoshgoftaar 2019)

It is noted that to build useful Deep Learning Models, the validation errors must be decreased continuously against the training errors. After submitting the final model, it was noted that Data Augmentation is a powerful method to overcome this issue. Not only that but also there are other techniques such as Dropout, Batch normalization, Transfer learning, Pretraining, On shot and Zero shot learning to be considered in finding high accuracy in Machine Learning applications.

ACKNOWLEDGEMENT

I wholeheartedly thank Dr Mohamed Reda Bouadjenek for guiding and advising and the prestigious Deakin university in giving me this opportunity to participate in this challenge where I learned a lot about these remarkable technologies, Machine Learning, Deep Learning and their applications that are useful in day to day activities.

REFERENCES

Microsoft (2017) Windows Defender ATP machine learning: Detecting new and unusual breach activity, Microsoft, accessed 27 September 2022.
https://www.microsoft.com/security/blog/2017/08/03/windows-defender-atp-machine-learning-detecting-new-and-unusual-breach-activity/
Shorten C and Khoshgoftaar T M (2019) ‘A survey on Image Data Augmentation for Deep Learning’, Journal of Big Data 6,60:1-48, https://doi.org/10.1186/s40537-019-0197-0

Shorten C and Khoshgoftaar T M (2019) A comparison of overfitting and desired convergence [image], Journal of Big Data, accessed 02 October 2022.
https://doi.org/10.1186/s40537-019-0197-0

Phung VH and Rhee EJ (23 October 2019) Schematic Diagram of a Basic convolutional neural network (CNN) architecture [image], Applied Sciences, accessed 02 October 2022. Available under Attribution - CC BY.
https://www.researchgate.net/publication/336805909_A_High-accuracy_Model_Average_Ensemble_of_Convolutional_Neural_Networks_for_Classification_of_Cloud_Image_Patches_on_Small_Datasets

IBM (2020) Recurrent Neural Networks, IBM, accessed 30 September 2022.
https://www.ibm.com/cloud/learn/recurrent-neural-networks
Radhakrishnan P (29 September 2017) ‘Image Captioning for Deep Learning’, Towards Data Science, accessed 1 October 2022.
https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2