Deep Learning (COSC 2779) – Assignment 1 – 2021

Karthi Narendrababu Geetha (s3835901)

1 Problem Statement

The lack of temporal information in still images is a major issue in recognizing human actions [2]. Its potential application entails image indexing, image and video analysis, human-computer interactions, etc [1, 3]. The problem statement involves identification of actions and general class of the corresponding human activity by employing deep neural network. The approaches and judgements on solving the above problem, alongside of its evaluation and results are discussed in this paper.

2 Dataset Provided

The dataset comprises of a directory containing 9532 images and 2 csv files, wherein one csv file is labelled.

- The labelled data containing 3031 rows of action and action classes is utilized to train the model.
- The unlabelled data consist of 2101 rows and is used to test the model's performance.
- There are 21 unique human actions and 5 generalized classes of their corresponding actions.
- Though the action and the action class have equivalent number of images, a slight imbalance is observed. Each action classes comprises of around 90 – 190 images.

3 Evaluation Framework

3.1.1 Loss function - Categorical Cross Entropy:

Categorical cross-entropy is the commonly used loss function for the multi-class classification problem. This function will provide the output in a probability of classes for each image.

3.1.2 Batch Size:

Batch size chosen for this problem is 32 images per batch, the smaller batch size improves the model's performance.

3.1.3 Epochs:

Number of epochs used in this problem is around 15 to 25. It is identified that the model is not being improved after 20 epochs.

3.1.4 Optimizer:

Adam optimizer is one of the best optimizers for the image classification problem and hence the Adam optimizer is chosen for this problem.

4 Approach & Justifications

The problem statement is approached by building neural network models and determine the model that provides an outstanding result.

4.1 Custom Data Generator:

To optimise the usage of system memory, the images are loaded in batches to train the model. Although the data generators are built using the ImageDataGenerator from Keras package, it does not support complex models such as multi-output model. The custom data loader is more powerful and flexible than the ImageDataGenerator class provided by Keras [11, 9]. The data loader reads the images, provides augmentation if necessary, and provides output images in the form of numpy array, in addition to the action and action class labels in an array.

4.2 Selection of Architecture:

4.2.1 Multi-output model:

Initially, a custom network for multi-output model is created to test its working. The model comprises of 3 convolution layers for each of the output classes along with the max pooling, dropout and batch normalization [5]. The two different output dense layers will have 'SoftMax' activation function as the model's output will be a classification of many classes.

4.2.2 VGG19 model:

VGG19 is the next model created, since the multi-output model does not improves the accuracy and loss. VGG19 model consisted of 16 convolution layer, 3 fully connected layers, 5 max pooling layers and 1 SoftMax activation function [6]. Along with all the layers, two output dense layers with 'SoftMax' activation function is appended.

4.2.3 Custom RESNET model:

Though, the total loss is minimized to 3.07%, the model is further trained to reduce the losses. Currently, the deep neural network provides promising results for image recognition problems. Therefore, Residual Network model is developed wherein each residual block consisting of 2 convolution layers with 'ReIU' activation function and batch normalization [8]. Higher the number of residual blocks utilized the more promising result is obtained. Thereby, a network with 3 groups of residual blocks, with 3 residual blocks in individual groups is developed [8].

RESNET101 Keras:

The loss percentage in the custom Residual network model is inflated when compared to VGG19 model. Hence, even deeper network is built with Keras application RESNET101 package. This network architecture embodies 345 layers alongside of Global Average Pooling layer and the output dense layers are appended. This architecture is chosen for the above problem, as a result the overall loss of the classification problem is reduced to 1.49%.

5 Experiments & Tuning

From the EDA analysis, it is clearly visible that there is major chance of data imbalance between the classes. For example, from the *figure 1a*, the image file count for the actions such as washing dishes, rowing a boat, cutting vegetables, taking photos is much lesser than the actions riding a horse, jumping, climbing etc. Similarly for the action classes the playing musical instruments has much lower data. The number of experiments performed is shown in the *table 1* alongside of the output obtained.

5.1 Data Augmentation:

In order to prevent class imbalance, the data augmentations were done such as random rotation, random flips and random shifts. Initially, each model is trained without augmentation, resulting in overfitting of the model. However, when the model is trained with data augmentation, some model exhibited slight underfitting and some were fitted precisely.

5.2 Transfer Learning:

With the help of pre-trained models such as RESNET101 and data augmentation, the training indicated reduced losses, but does not improve the accuracy of the model. Hence, the saved weights from the 'imagenet' dataset are used in order to implement transfer learning, which in turn improved the accuracy of the model (*figure 2a, 2b*).

5.3 Class Reweighting:

Class reweighting is the concept of influencing the loss function to set relatively higher cost to the data from minority classes [11]. It is done with the help of compute_class_weight function from the scikit-learn package, which computes the class weights using *n_samples / (n_classes * np.bincount(y))*. Class reweighting improved the prediction rate for the imbalance classes as well, however the overfitting problem still existed.

5.4 Learning Rate Scheduler:

With the help of Learning Rate scheduler, the learning rate of optimizer is modified during the training process. The scheduler works purely based on the loss function values from each epochs [11]. With the use of scheduler, the training performance is improved where the accuracy is almost 100% for the model. However, model still needs to be fine-tuned.

5.5 Re-training of pre-trained network:

In order to enhance the performance of the pre-trained models, the models were trained again by setting trainable to False for some top-level layers. Eventually, the weights of the pre-trained model is modified with respect to the provided dataset. Setting trainable to False after 150 layers slightly improved the performance of model.

5.6 Dropouts:

Dropout is a regularization technique where the randomly selected convolutional neurons will be ignored during the training process, which helps the model to be more generalized. Initially, the models were trained with 60% of the dropout rate, and then reduced to 20% dropout rate which improved the performance of the model.

6 Ultimate Judgment, Analysis & Limitations

6.1 Final model:

The chosen model architecture is RESNET101 with 20% dropout rate. The model is compile with data augmentations, class reweighting and learning rate scheduler. The model provides the accuracy of around 70% for the actions and 85% for the action classes, also the overall loss is even reduced to around 1.5%. It is also generalized with low bias and variance, which provides a better recall rate for all the classes when compared to other models.

6.2 Limitations:

Since the model is more focused on reducing the problem of class imbalance, the prediction done for the minority class is accurate than the other classes.

7 References

1. Stanford 2020, Stanford 40 Actions, viewed 12 September 2021, <http://vision.stanford.edu/Datasets/40actions.html>.
2. Zhang, J., Lin, H., Nie, W., Chaisorn, L., Wong, Y. and Kankanhalli, M., 2015, Human Action Recognition Bases on Local Action Attributes, Journal of Electrical Engineering and Technology, vol. 10, pp.64-74.
3. Ma, W, and Liang, S., 2021, Human-Object Relation Network for Action Recognition in Still Images, Action recognition in still images, Tongji University, vol, 4, pp. 2-6.
4. Zhao, Z., Ma, H, and You, S., 2021, Single Image Action Recognition using Semantic Body Part Actions, Semantic part action as mid-level semantics, Australia National University, vol. 5, pp.2-7.
5. Medium 2021, Building a multi-output Convolutional Neural Network with Keras, viewed 3 September 2021, <https://towardsdatascience.com/building-a-multi-output-convolutional-neural-network-with-keras-ed24c7bc1178>.
6. Team, K., 2021, Keras documentation: VGG16 and VGG19, viewed 8 September 2021, <https://keras.io/api/applications/vgg/>.
7. Journal of Xidian University 2021, Image Recognition using Deep Learning Techniques, vol.15, pp. 2-9.
8. Medium, 2021, Building a ResNet in Keras, viewed 5 September 2021, <https://towardsdatascience.com/building-a-resnet-in-keras-e8f1322a49ba>.
9. TensorFlow. 2021, tf.keras.applications.resnet.ResNet101|TensorFlow Core v2.6.0, viewed 7 September 2021, <https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/ResNet101>.
10. Medium 2021, Write your own Custom Data Generator for TensorFlow Keras, viewed 10 September 2021, <https://medium.com/analytics-vidhya/write-your-own-custom-data-generator-for-tensorflow-keras-1252b64e41c3>.
11. Tandia, F., 2021, Some Tricks for Handling Imbalanced Dataset (Image Classification), viewed 8 September 2021, <https://www.linkedin.com/pulse/some-tricks-handling-imbalanced-dataset-image-m-farhan-tandia/>.

8 Appendix

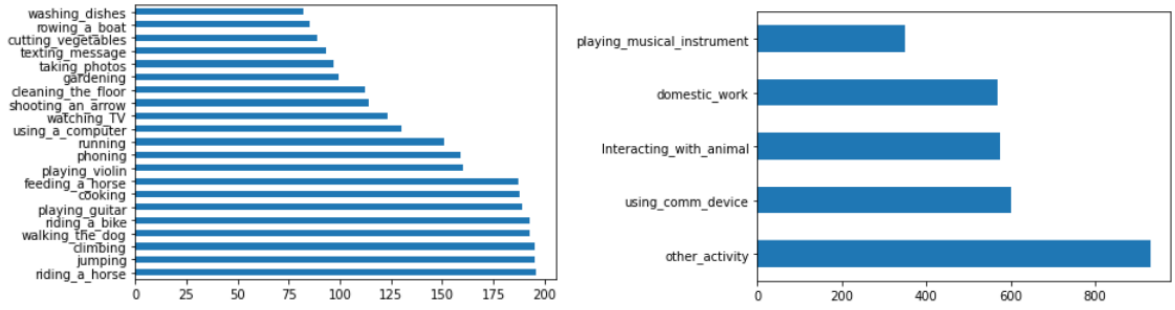

Figure 1: understanding the problem of class imbalance by doing EDA analysis

model	weights	augment	class_weight	schedule	action_loss	action_class_loss	action_acc	action_class_acc	loss
multi-output model	FALSE	FALSE	FALSE	FALSE	4.0449	3.1422	0.2639	0.4358	10.7806
multi-output model	FALSE	TRUE	FALSE	FALSE	5.8683	5.4484	0.1962	0.401	16.975
vgg19	FALSE	TRUE	FALSE	FALSE	2.8892	1.457	0.1198	0.3854	6.5517
vgg19	TRUE	TRUE	FALSE	FALSE	1.3722	0.6458	0.6528	0.7986	3.0765
resnet	FALSE	FALSE	FALSE	FALSE	4.0204	2.7663	0.1285	0.3507	48.514
resnet	FALSE	TRUE	FALSE	FALSE	2.86	1.532	0.2465	0.4549	32.3327
resnet	FALSE	TRUE	TRUE	FALSE	3.185	1.9085	0.2639	0.441	26.8049
resnet	FALSE	TRUE	TRUE	TRUE	2.6863	1.4698	0.2049	0.4497	26.3649
resnet101	FALSE	FALSE	FALSE	FALSE	2.8355	1.4296	0.1389	0.3924	6.4415
resnet101	FALSE	TRUE	FALSE	FALSE	2.5915	1.347	0.2101	0.4566	5.9257
resnet101	FALSE	TRUE	TRUE	FALSE	2.538	1.3228	0.2552	0.4965	3.5601
resnet101	FALSE	TRUE	TRUE	TRUE	2.5024	1.2707	0.2396	0.4965	3.4679
resnet101	TRUE	TRUE	FALSE	FALSE	1.0485	0.5128	0.6719	0.8316	2.3936
resnet101	TRUE	TRUE	FALSE	FALSE	0.9655	0.5299	0.6892	0.8212	2.295
resnet101	TRUE	TRUE	FALSE	TRUE	1.6027	0.7325	0.6736	0.8507	3.5483
resnet101	TRUE	TRUE	TRUE	TRUE	1.3237	0.623	0.6597	0.8056	1.8217
resnet101-retrain 100 layers	TRUE	TRUE	FALSE	FALSE	1.1369	0.5139	0.7396	0.8576	2.5252
resnet101-retrain 100 layers	TRUE	TRUE	FALSE	TRUE	1.397	0.6738	0.6007	0.7864	3.1509
resnet101-retrain 100 layers	TRUE	TRUE	TRUE	TRUE	1.6562	0.8095	0.5434	0.7083	2.3006
resnet101-retrain 150 layers	TRUE	TRUE	FALSE	FALSE	1.1259	0.6769	0.7188	0.842	2.7556
resnet101-no dropout	TRUE	TRUE	FALSE	FALSE	0.9757	0.4701	0.6892	0.8351	2.2209
resnet101-no dropout	TRUE	TRUE	TRUE	TRUE	1.5247	0.7063	0.6632	0.8403	2.0788
resnet101-20% dropout	TRUE	TRUE	FALSE	FALSE	0.974	0.4658	0.6788	0.8368	2.2117
resnet101-20% dropout	TRUE	TRUE	TRUE	TRUE	1.1512	0.556	0.6892	0.8455	1.6103

Table 1: Experiments done to identify the perfect model for the problem statement

Classification Report for Actions

	precision	recall	f1-score	support
walking_the_dog	0.12	0.14	0.13	21
riding_a_bike	0.25	0.45	0.33	33
gardening	0.17	0.26	0.20	35
cooking	0.00	0.00	0.00	20
jumping	0.24	0.24	0.24	42
cutting_vegetables	0.16	0.43	0.23	14
watching_TV	0.21	0.08	0.12	36
cleaning_the_floor	0.13	0.19	0.15	27
shooting_an_arrow	0.17	0.16	0.16	32
texting_message	0.20	0.02	0.04	41
playing_violin	0.37	0.50	0.43	40
feeding_a_horse	0.21	0.57	0.31	28
taking_photos	0.17	0.24	0.20	17
washing_dishes	0.25	0.18	0.21	22
riding_a_horse	0.33	0.05	0.08	21
rowing_a_boat	0.50	0.09	0.15	22
playing_guitar	0.00	0.00	0.00	14
climbing	0.11	0.08	0.09	26
running	0.25	0.16	0.20	37
phoning	0.12	0.05	0.07	22
using_a_computer	0.20	0.27	0.23	26
accuracy			0.21	576
macro avg	0.20	0.20	0.17	576
weighted avg	0.21	0.21	0.18	576

Classification Report for Action Class

	precision	recall	f1-score	support
Interacting_with_animal	0.38	0.35	0.36	107
other_activity	0.42	0.24	0.31	112
domestic_work	0.53	0.81	0.64	169
using_comm_device	0.35	0.08	0.13	73
playing_musical_instrument	0.39	0.47	0.43	115
accuracy			0.45	576
macro avg	0.41	0.39	0.37	576
weighted avg	0.43	0.45	0.42	576

Figure 2a: Classification report at the initial stage of the experiment

Classification Report for Actions

	precision	recall	f1-score	support
walking_the_dog	0.86	0.83	0.84	23
riding_a_bike	0.83	0.85	0.84	34
gardening	0.64	0.64	0.64	36
cooking	0.52	0.63	0.57	19
jumping	0.91	0.82	0.86	39
cutting_vegetables	0.42	0.94	0.58	16
watching_TV	0.66	0.55	0.60	38
cleaning_the_floor	0.41	0.44	0.42	25
shooting_an_arrow	0.96	0.76	0.85	33
texting_message	0.67	0.76	0.72	38
playing_violin	0.80	1.00	0.89	37
feeding_a_horse	0.84	0.96	0.90	27
taking_photos	1.00	1.00	1.00	16
washing_dishes	0.52	0.63	0.57	27
riding_a_horse	0.71	0.57	0.63	21
rowing_a_boat	0.64	0.41	0.50	22
playing_guitar	0.17	0.21	0.19	14
climbing	0.50	0.64	0.56	25
running	0.85	0.63	0.72	35
phoning	0.60	0.14	0.23	21
using_a_computer	0.83	0.63	0.72	30
accuracy			0.69	576
macro avg	0.68	0.67	0.66	576
weighted avg	0.71	0.69	0.68	576

Classification Report for Action Class

	precision	recall	f1-score	support
Interacting_with_animal	0.80	0.89	0.85	101
other_activity	0.82	0.90	0.85	115
domestic_work	0.91	0.80	0.85	173
using_comm_device	0.89	0.83	0.86	71
playing_musical_instrument	0.80	0.82	0.81	116
accuracy			0.84	576
macro avg	0.84	0.85	0.84	576
weighted avg	0.85	0.84	0.84	576

Figure 2b: Final Classification report with the chosen model

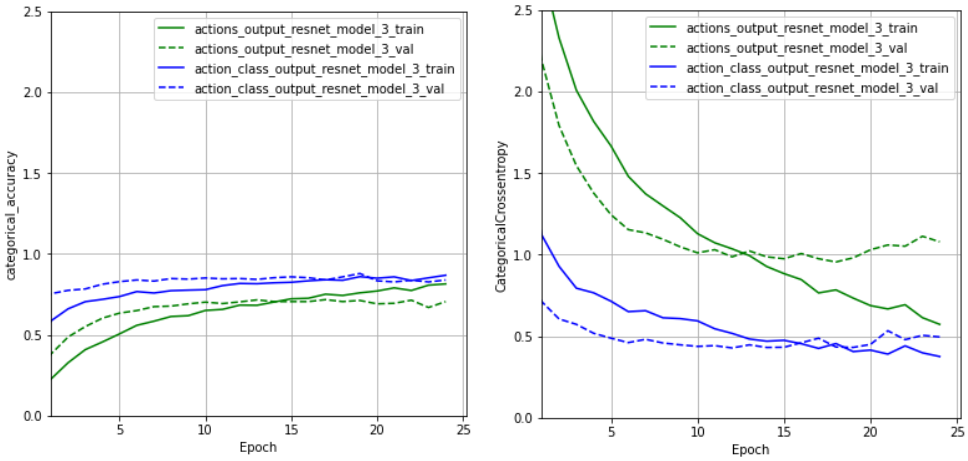

Figure 3: Accuracy and loss plots for the chosen model