

A Report on

Human Action Recognition using Self Attention Bidirectional and plain Bidirectional LSTM model

By-
Soumyajit Karmakar
BTech CSE

At
CSIR- CEERI Pilani



भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Information Technology Guwahati



सीएसआईआर-केन्द्रीय इलेक्ट्रॉनिकी अभियांत्रिकी अनुसंधान संस्थान

CSIR - CENTRAL ELECTRONICS ENGINEERING RESEARCH INSTITUTE


(विज्ञान तथा प्रौद्योगिकी मंत्रालय, भारत सरकार/MINISTRY OF SCIENCE & TECHNOLOGY, GOVT. OF INDIA)

पिलानी, राजस्थान (भारत)/Pilani, Rajasthan - 333031 (INDIA)


दिनांक Date 31/12/2021

Certificate

This is to certify that Mr. Soumyajit Karmakar, B.Tech (Computer Science), 3rd year student of Indian Institute of Information Technology (IIIT), Guwahati did one-month internship training during the period of 29th November to 31st December, 2021 under our guidance, on the topic, "Human Action Recognition using Self Attention Bidirectional and plain Bidirectional LSTM Model".


31/12/2021

Dr. Sanjay Singh
Principal Scientist
Intelligent Systems Group
CSIR-CEERI Pilani


31/12/2021

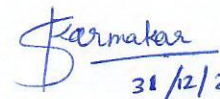
Mr. Sumeet Saurav
Senior Scientist
Intelligent Systems Group
CSIR-CEERI Pilani

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude towards all the people and my organizations that helped me in completing this internship successfully.

I would like to thank CEERI Pilani and my college IIITG for giving me this opportunity to do Winter Internship at such a prestigious organization.

I would sincerely like to thank my mentors Mr. Sumeet Saurav and Dr. Sanjay Singh at Intelligent Systems Group, CSIR-CEERI, Pilani, for guiding me through this internship, providing useful resources and helping me understand the project domain.


31/12/2021

(Soumyajit Karmakar)

ABSTRACT

Video action recognition is a very important task for many of the modern activities such as automated surveillance, self-driving cars, elderly behaviour monitoring, human-computer interaction, content-based video retrieval, and video summarization. In this domain Recurrent Neural Networks (RNN) have found immense success. In particular Long Short Term Memory (LSTM), a type of RNN, prove to be exceptional at processing sequential data, such as a video.

In this project I tried to implement a Human Action Recognition Model using pretrained CNN models such as ResNet, DenseNet and MobileNet to extract features and have used a Deep Bidirectional LSTM with self attention to process the data and make the predictions. I have tested my model using the datasets UCF11, UCFSports and JHMDB. Many of the results I got were very close to the State-of-the-Art results.

TABLE OF CONTENT

	Heading	Page No.
0.	Acknowledgement	i
0.	Abstract	ii
0.	Table of Content	iii
1.	Introduction	1
2.	Literature Survey	2
3.	Dataset Description	3
4.	CNN Models	4 – 7
5.	LSTM Designs	8 – 9
6.	Model Details	10
7.	Results	11 – 19
8.	Analysis	20
9.	Summary	21
10.	Conclusion	22
11.	References	23

1. INTRODUCTION

In today's age as we try to automate more and more actions, the need for having computer vision is felt more than ever. The vision is one of the 5 fundamental senses, arguably the most important one. But giving computers the ability to capture an image and understand it is no easy task. There have been numerous researches and breakthroughs in the past couple of decades on this topic. In this project I have tried to implement a model which can give State of the Art results as of today.

The process of decoding an image by the computer can be loosely broken into two sub tasks:

- i) Feature Extraction,
In this step the input image is converted into a form which is more manageable and easier to work with. This step is done by the Convolutional Neural Network (CNN).
- ii) Processing the Feature Set,
In this step the feature set is processed to predict various properties about the image. This step is done by the Recurrent Neural Network (RNN).

Both CNN and RNN are types Neural Networks both with different uses. The CNN is used when we have to process and convert huge data stored in grid like topology into much smaller and manageable size. The RNN is used when we need to process sequential data to make some predictions.

The entire project is done in Python language.

2. LITERATURE SURVEY

In this project I have tried to implement the model discussed in the paper [1] by Amin Ullah et al. The model is trained and tested on the datasets UCF11, UCFSports and JHMDB. I further improved the model based on the paper [2] by Khan Muhammad et al.

In the first paper the authors used AlexNet Convolutional Neural Network (CNN), which was trained on the ImageNet dataset, to extract features and Deep Bidirectional Long Short Term Memory (DB-LSTM) to process the features. In this paper to get rid of some redundant frames and improve the efficiency of the model they jump 6 frames when processing each video, with this they could achieve an accuracy of 92.84% on YouTube Actions dataset which is 1.24% better than the next best, 87.64% on HMDB51 dataset which is 19.14% better than the next best and 91.21% on UCF11 dataset which is 2.11% better than the next best.

In the second paper the authors used Self Attention based DB LSTM model to make the predictions. For the feature extraction part, they tried to improve upon the existing CNN models by implementing a Dilated CNN (DCNN) with skip connections. They improved upon the loss function by introducing a Centre Loss Function, which they could prove to perform better than the traditional loss function for action recognition. With all these new modifications they were able to achieve a solid accuracy of 99.10% on the UCFSports dataset which is 0.5% better than the next best, 98.30% on the UCF11 dataset which is 1.4% better than the next best and 80.20% on the JHMDB dataset which is 3.9% better than the next best.

3. DATASET DESCRIPTION

1. UCF11

It is the biggest dataset, as compared to the other two, it has a total of 1600 videos with a total of 11 different classes such as shooting, jumping, swimming etc. All the videos are around 5 to 8 seconds long with 30 frames each second. Variation in illumination and cluttered background make this a fairly challenging dataset to work with.

2. UCFSports

This dataset contains only 150 sequences with a total of 10 different classes such as skateboarding, golf swing, lifting, etc. The duration of the videos is under 10 seconds with 10 frames per second. This dataset is composed from various sources like the BBC and ESPN. This dataset contains videos taken from various angle, for example it contains many videos of the golf swing action taken from three different angles front, back and side.

3. J-HMDB

This dataset contains 923 videos with 21 classes. It contains everyday activities such as clapping, brushing hair, catching, etc. The duration of each video in this dataset is the shortest among the three, at around a couple of seconds each. The frame rate here is 30 frames per second. The diversity of the data in this dataset makes it challenging for any classification task.

4. CNN MODELS

A Convolutional Neural Network (CNN) is a type of neural network which specialises in processing data that has a grid-like topology, such as an image, audio, etc. A single image is made up of thousands of pixel values, to process it with a basic neural network we have to arrange all the pixel values in an array and feed it in. This is a very computationally expensive task since even a moderately detailed image nowadays can have more than a million pixel values, which will result in a huge number of trainable weights values. Also, it is observed that such dense connection where each pixel value is fed to every other neuron degrades the performance of the model.

In a CNN architecture, each layer is assigned a filter matrix. Each neuron in a layer looks only to a small part of the output of the previous layer (input image if it is the first layer) and applies the filter to that part only. The filter can be thought as a weight matrix which is multiplied with the pixel values in a small region of the image, then added together to get a single value as the output of that neuron.

This approach solves both the problems, firstly it drastically reduces the number of trainable weights, and secondly as the model is not densely connected, it improves the performance.

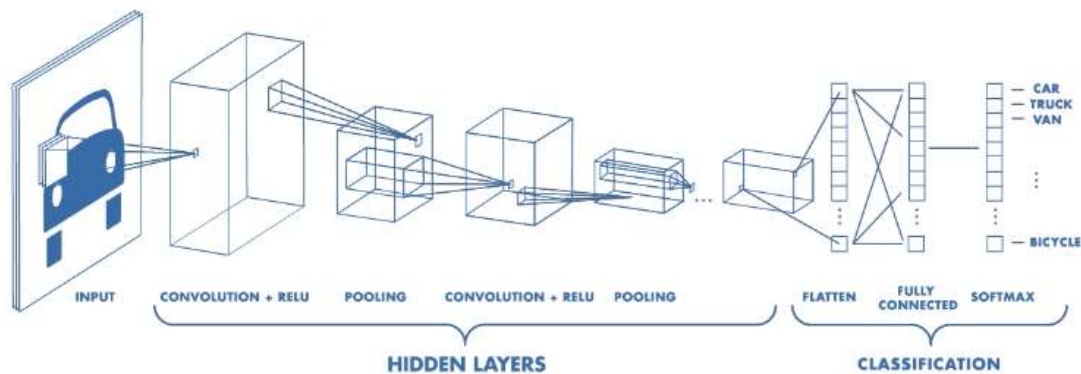


Fig 1: Basic CNN architecture, Ref. [8]

Initially an image has 3 channels, for the three primary colours. In the first CNN layer the top left neuron will look only to the top left area of the image with size equal to the filter size. It will perform element wise multiplication with all the pixels across all the channels, and then add all the multiplied values to generate a single output. For the next neuron on the right the window will shift to the right and the process is repeated. Using one filter will result in an output space of one channel only, using n filters in a layer will result in n channels in the output of that layer.

After passing through many layers or when the image has been reduced to the required size, we can flatten the resulting 3D matrix into a 1D array which we can further pass into a Dense Neural Network or pass it into some other processing unit as required.

Training a CNN model requires a huge amount of training data, computing power and time. Thus, training a CNN model from scratch is not feasible for projects such as mine.

For this project I have chosen 3 pretrained CNN models the ResNet50, the DenseNet121 and the MobileNetV2.

1. ResNet50:

A ResNet [3] or Residual Network is basically a deep CNN but the addition of Residual Blocks.

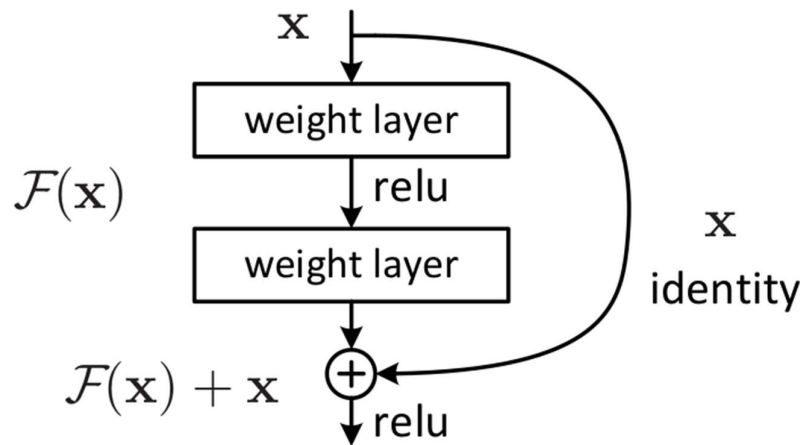


Fig 2: Residual Block diagram, Ref. [9]

These Residual Blocks provide skip connections which provide a path which gives a short cut path for the output of one layer to skip past the next few layers.

In Deep Neural Networks it is observed that increasing the depth of the network doesn't always improve the performance, often times deep networks would suffer from Vanishing Gradient problem wherein the loss which is calculated at the end of the network after making the prediction is not able to travel all the way back to the earlier layers to update their weights during the backpropagation stage. The second problem with training the deeper networks is, performing the optimization on huge parameter space and therefore naively adding the layers leading to higher training error.

Using residual blocks overcomes these problems and allows the training deeper networks.

There are many different variations of the ResNet architecture, the one I am using is ResNet50, here 50 implies that this model has 50 layers.

2. DenseNet121:

A DenseNet [4] or Densely Connected Convolutional Network provides another method for solving the problems of deep neural networks. Here instead of Residual Blocks, each layer is directly connected to each other layer. It can be shown that in

ResNets many layers are redundant and can be dropped all together, which can also help in reducing the total number of parameters. The Vanishing Gradient issue is not a problem at all since each layer has direct access to the gradients from the loss function at the end of the network and the input at the start of the network.

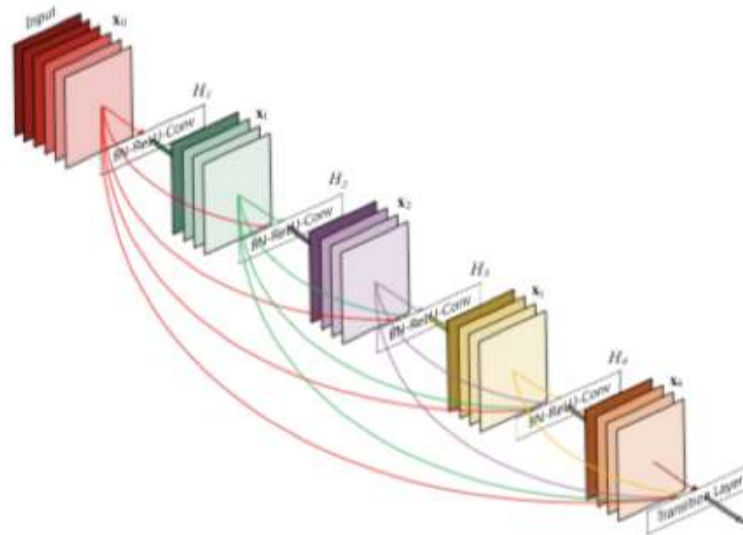


Fig 3: DenseNet architecture, Ref. [10]

The DenseNet architecture has many variants as well, I will be using the DenseNet121 where 121 implies it has 121 trainable layers in total.

3. MobileNetV2

The MobileNet [5] is a small, low-latency, low-power CNN model with the primary objective to bring the power of deep neural networks on mobile devices. MobileNet uses Depth-wise Separable Convolutions instead of standard convolution to reduce model size and computation. Here instead of the traditional convolution step, it is broken in two separate steps, ie., depth wise convolution and point wise convolution. This method reduces the number of parameters dramatically.

The unique property of MobileNet is that it allows the user to control the depth and size of the network, so for low power devices one may choose smaller network for low computational costs, while for larger systems one can use bigger version of the network which takes more computational power but gives better results.

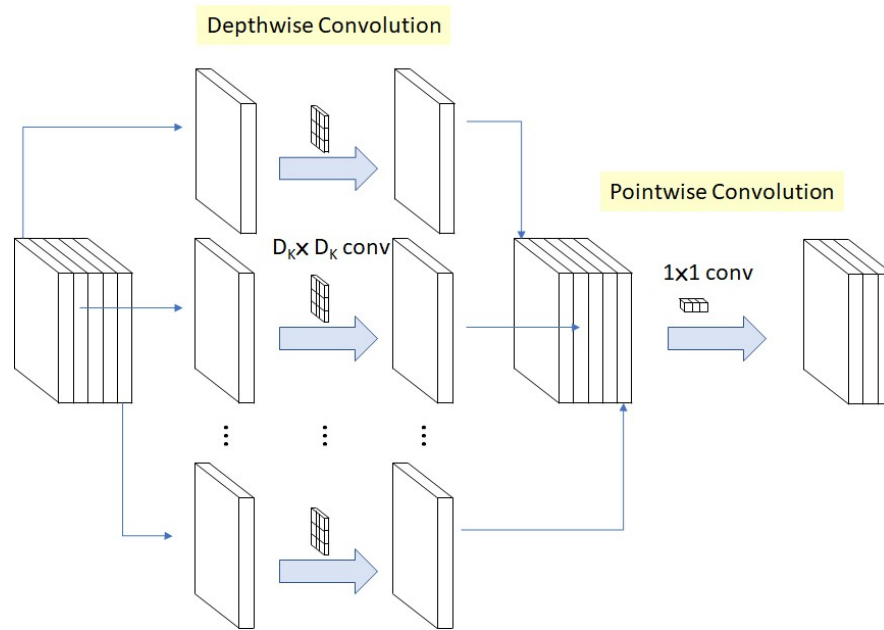


Fig 4: Depth-wise Separable Convolution working, Ref. [11]

The MobileNetV2 is the improved version of the original MobileNet architecture. It has two new features, linear bottleneck between the layers and skip connections between the bottlenecks. These additions can give about a 35% improvement in performance as compared to the original model while giving the same accuracy.

All of these CNN models have a pooling layer followed by a softmax classifier built in at the end. For my purpose I have to remove the last layer, the softmax unit, and replace it with the LSTM model to make the predictions.

5. LSTM DESIGNS

The core principle behind a RNN network is that instead of passing the entire input data to the neural network in one go, we input the data one by one sequentially, essentially bringing into play the time variable as well. If we have an array of input values, we feed the first value in the network and get some output, now for the next output we will feed the next input along with the previous output.

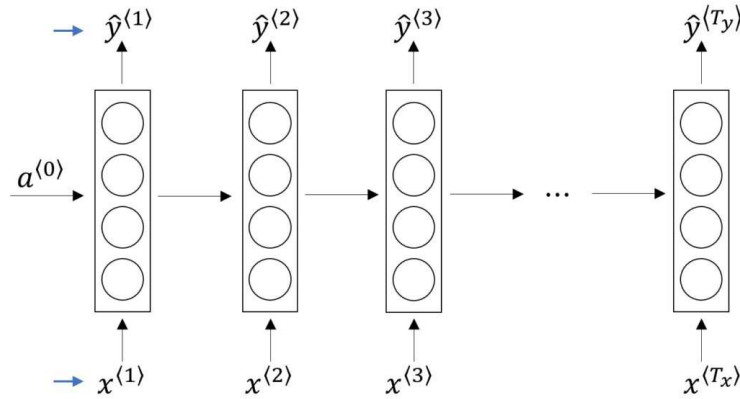


Fig 5: Basic RNN structure, Ref. [12]

During an epoch, for each input value we use the same set of weights for the neural network.

The first type of improvement can be done is that instead of flowing the data in only one direction from start to end, we may also feed the last input value first and reverse the flow of data. Performing both these flows together and combining their results gives us the Bidirectional RNN architecture.

Often times some information which is obtained in the earlier inputs needs to be preserved for some effects in the later input values. The basic RNN architecture is not suited for such needs. For this the Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) were introduced.

In an LSTM [6], along with the basic RNN block we add a memory cell and 3 new gates, to control the memory cell.

The memory cell is responsible for picking up some important data and retaining it until it is used to make some output later in the cycle. The three gates are:

- The Update Gate (aka Input Gate), it has its own weight matrix and activation function which it used to determine if the current input should be able to update the values the memory cell or not.
- The Forget Gate, it uses its own weight matrix and activation function to determine, using the current input, if the contents of the memory cell should be forgotten or not.
- The Output Gate, it uses another weight matrix and an activation function to determine how much should the memory cell affect the output for the current input.

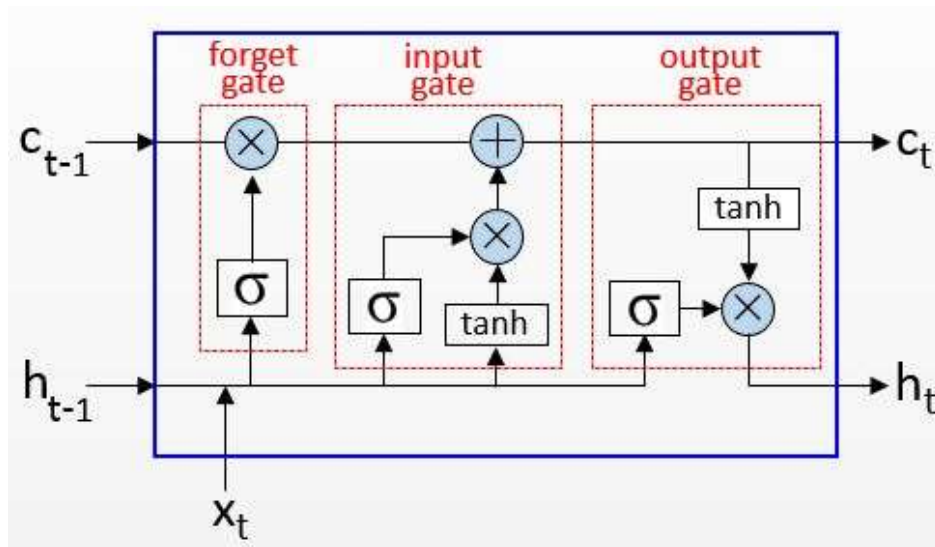


Fig 6: LSTM block diagram, Ref. [13]

The GRUs are basically a simplification of the LSTM model where there is only one gate which controls all the operations. This reduces the complexity and, in some cases, may also improve the performance.

Similar to the basic RNN idea we can easily implement Bidirectional LSTMs and GRUs as well, by flowing the data in both the directions.

The LSTM and GRU did indeed improve the performance for shorted sequences, but longer sequences, when there are multiple updates done on the memory cell, the performance would suffer again. For this the Self Attention Transformer [7] was introduced. Here instead of having a memory cell to carry all the information from the previous cells, we first run a basic RNN model over the entire sequence and then feed all of the outputs to another network, which now makes predictions based on the outputs of all the individual blocks.

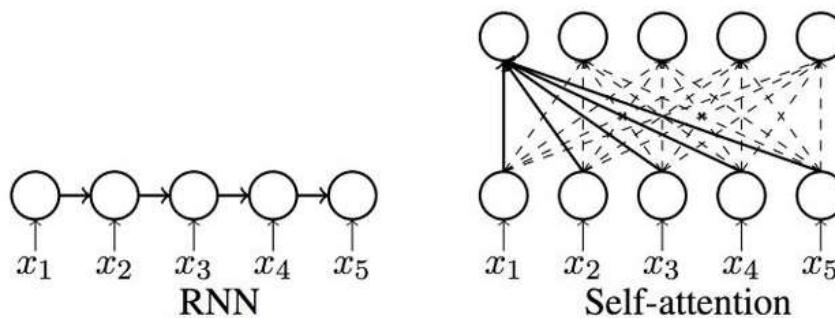


Fig 7: Self Attention model, Ref. [14]

6. MODEL DETAILS

To compare the various LSTM models, I tested 3 types, the plain LSTM architecture, the Bidirectional LSTM, and the Self Attention Bidirectional LSTM. I tested each of the three models using 1, 2, 3 layers and 32, 64, 128, 256 neurons in each layer. As the final layer of every model there is a softmax classifier that makes the predictions.

To summarise, for each of the three datasets I first passed it through the before mentioned three CNN models, then for each of the three sets of features I trained 3 different LSTM models, as discussed above, while varying the depth and number of neurons. In total $3 * 4$, i.e., 12 different models for each feature sets and so $3 * 12$, i.e. 36 different results for each dataset.

I have used the Keras API for testing all the models. For each of the test runs I have used a train test split of 80:20. I found experimentally most of the models converge at around 20 epochs, so I fixed the max epochs at 20. For the optimiser I have used the ADAM optimiser and the loss function being Categorical Crossentropy.

The final results are discussed in the next section.

7. RESULTS

1. UCF11 Dataset

ResNet50 CNN

Among all the models the best results were obtained for when a 2-layer Self Attention Bi-LSTM with 32 neurons in each layer was used. The accuracy was around 99.1% while the loss was 0.035. A similar model with 64 neurons each layer was a close second.

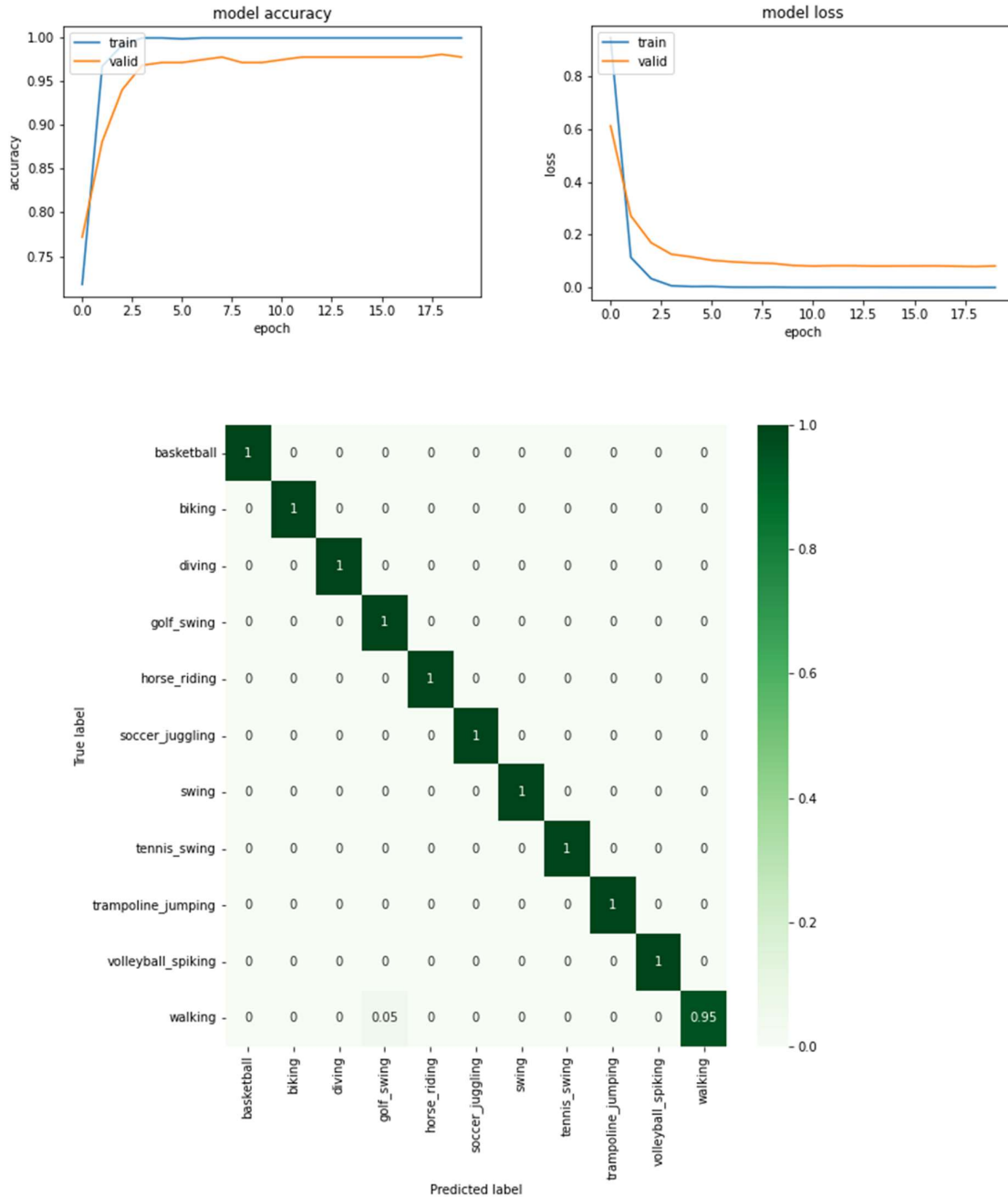


Fig 8: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix - ResNet50, UCF11 dataset

DenseNet121 CNN

Among all the models the best results were obtained for when a 1-layer Bi-LSTM with 32 neurons in each layer was used. The accuracy was around 99.3% while the loss was 0.037.

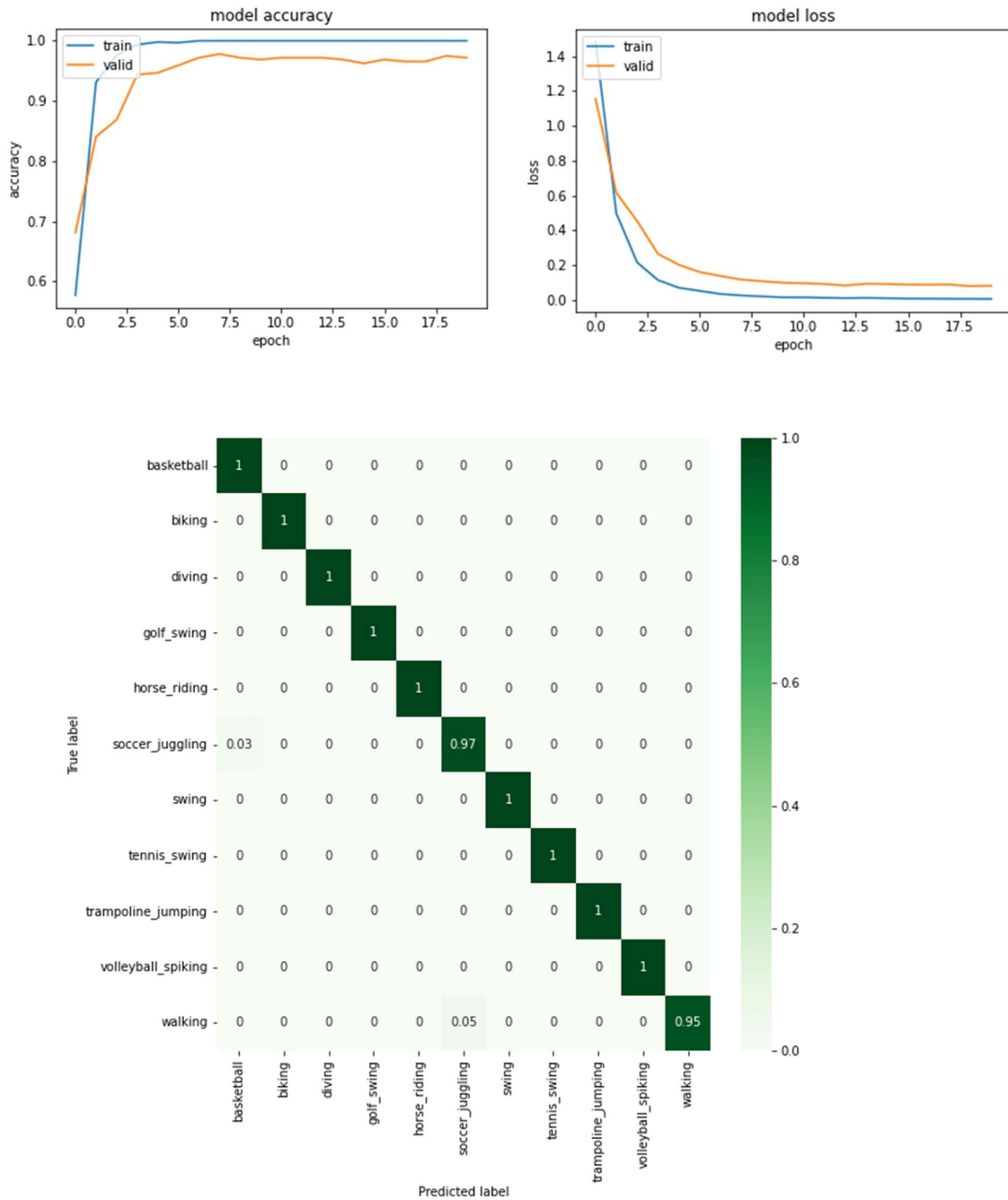


Fig 9: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – DenseNet121, UCF11 dataset

MobileNetV2 CNN

Among all the models the best results were obtained for when a 2-layer Bi-LSTM with 64 neurons in each layer was used. The accuracy was around 97.8% while the loss was 0.077.

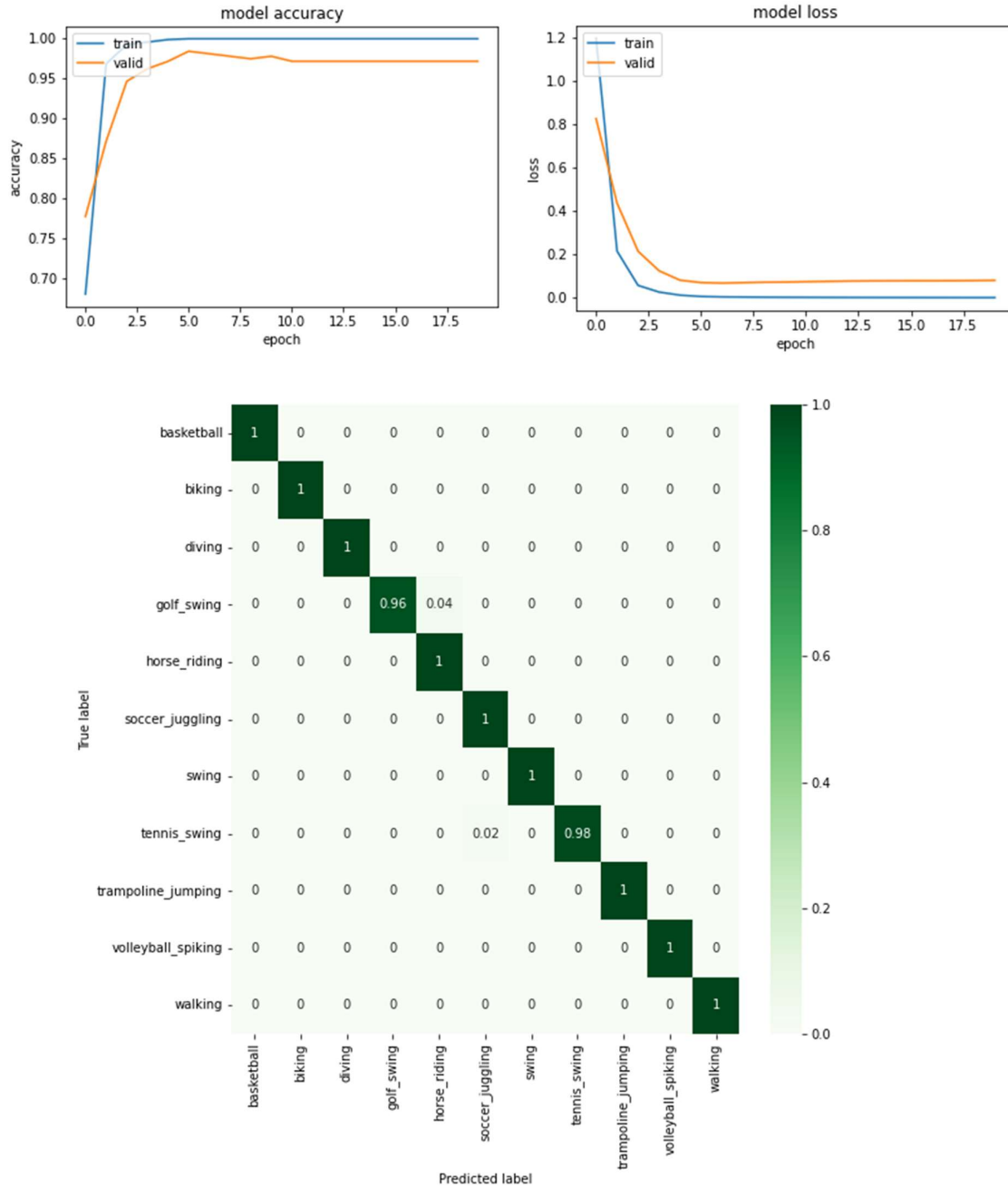


Fig 9: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – MobileNetV2, UCF11 dataset

2. UCFSports Dataset,

ResNet50 CNN

Among all the models the best results were obtained for when a 3-layer Self Attention Bi-LSTM with 128 neurons in each layer was used. The accuracy was around 83.3% while the loss was 1.24.

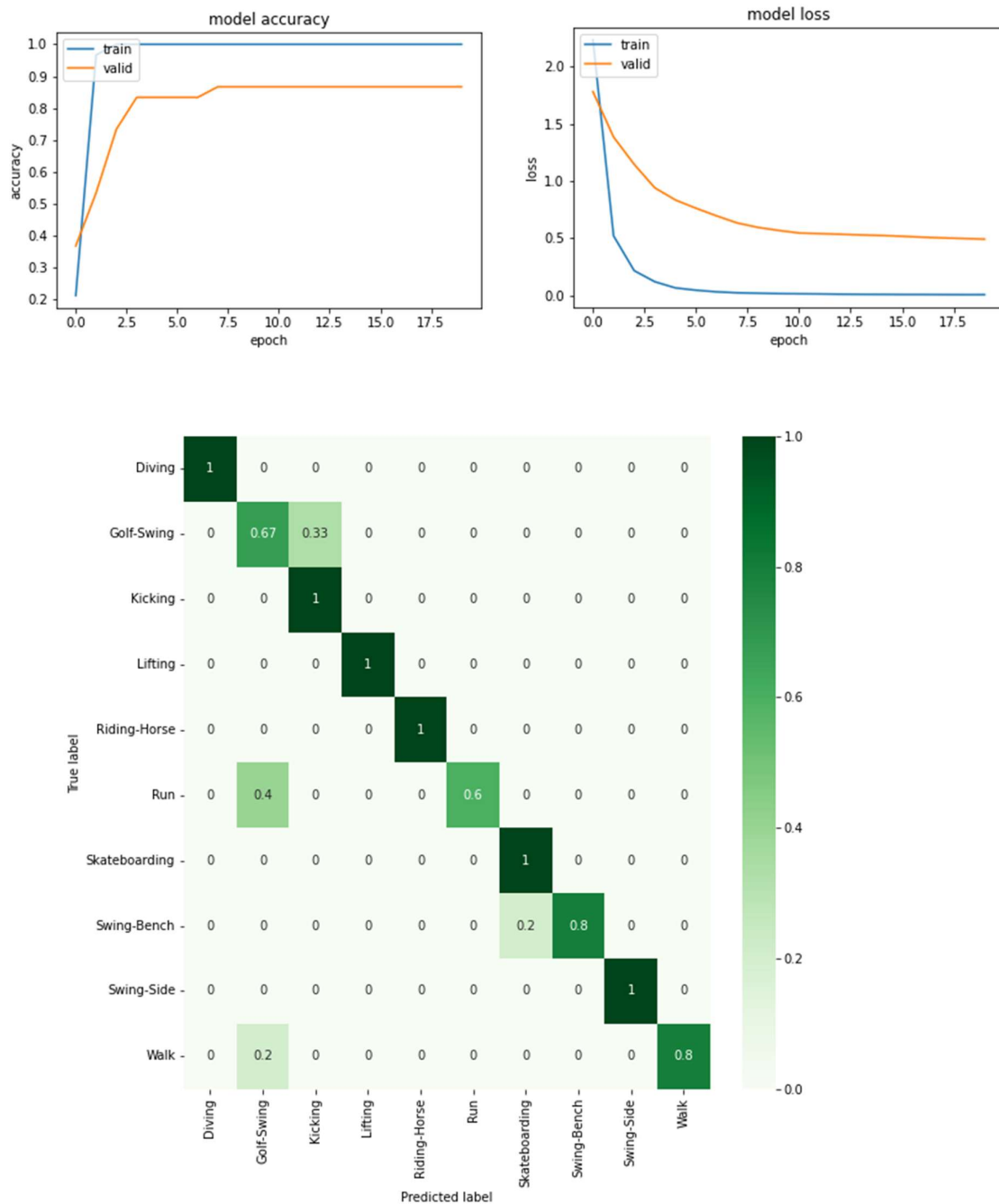


Fig 10: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – ResNet50, UCFSports dataset

DenseNet121 CNN

Among all the models the best results for when a 1-layer Bi-LSTM with 256 neurons in each layer was used. The accuracy was around 89.9% while the loss was 0.57.

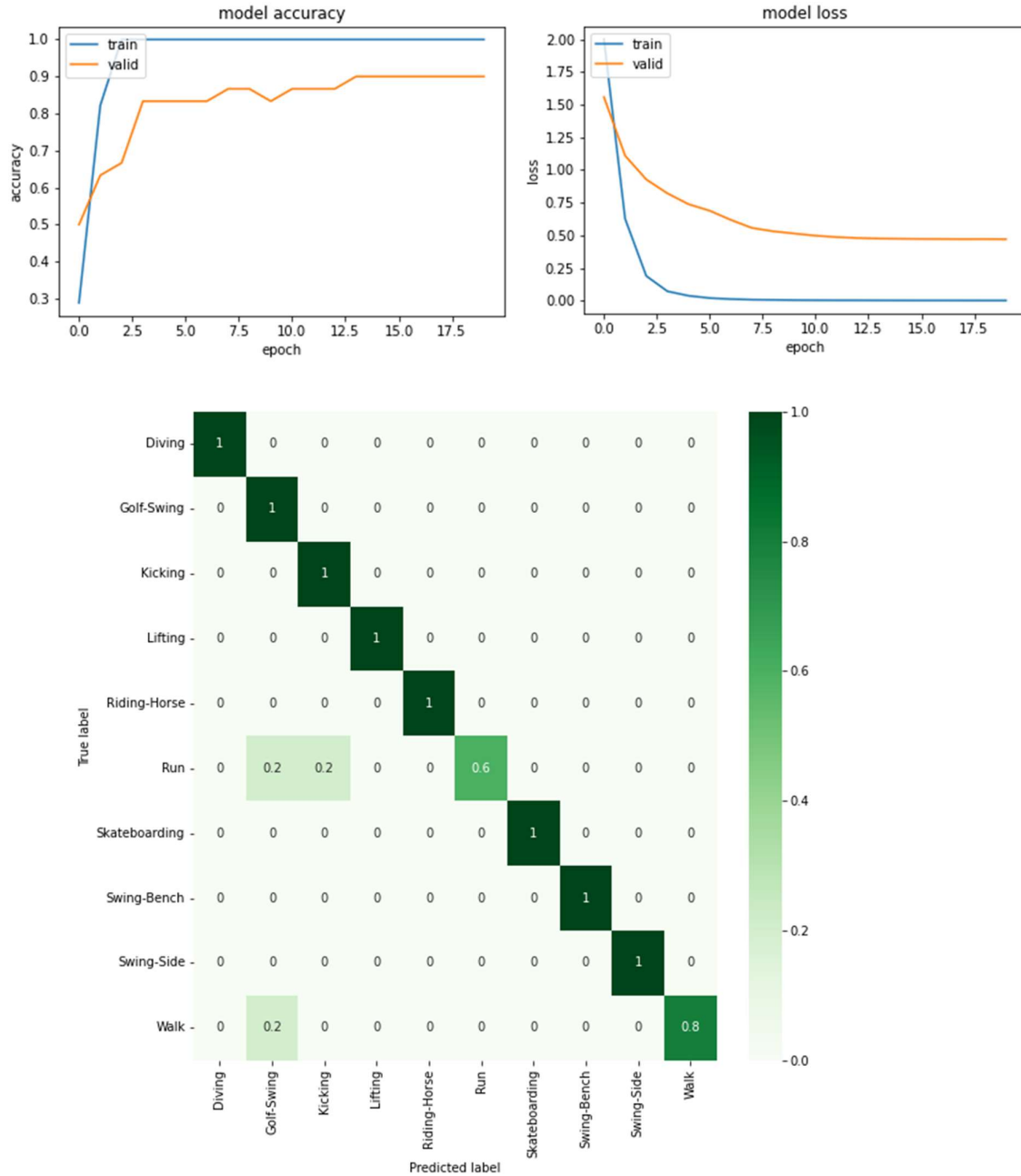


Fig 11: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – DenseNet121, UCFSports dataset

MobileNetV2 CNN

Among all the models the best results were obtained for when a 2-layer Bi-LSTM with 128 neurons in each layer was used. The accuracy was around 89.9% while the loss was 0.52.

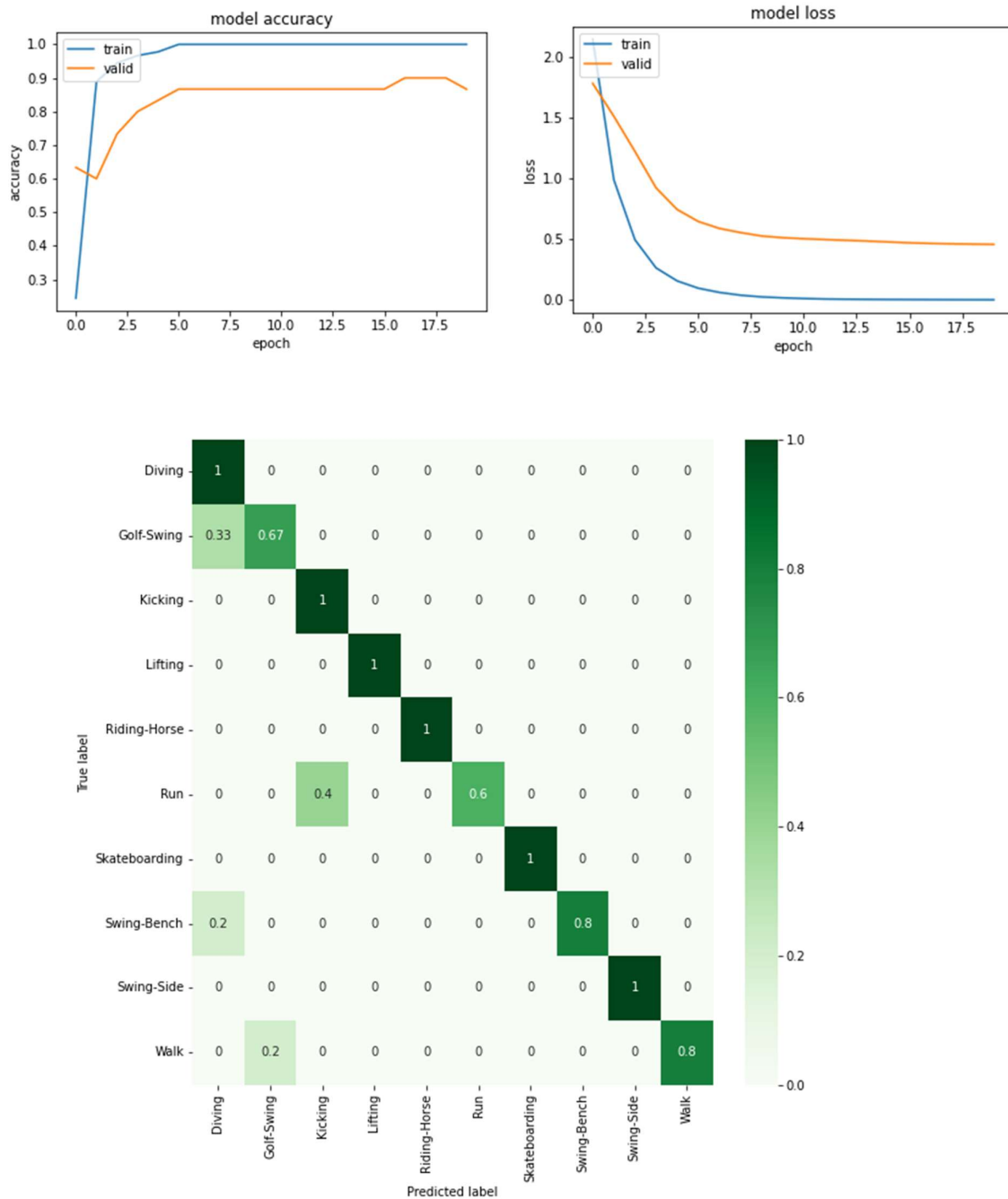


Fig 12: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – MobileNetV2, UCFSports dataset

3. JHMDB Dataset,

ResNet50 CNN

Among all the models the best results were obtained for when I used a 1-layer Self Attention Bi-LSTM with 128 neurons in each layer was used. The accuracy was around 73.1% while the loss was 1.43. A similar model with 32 in each layer was a close second.

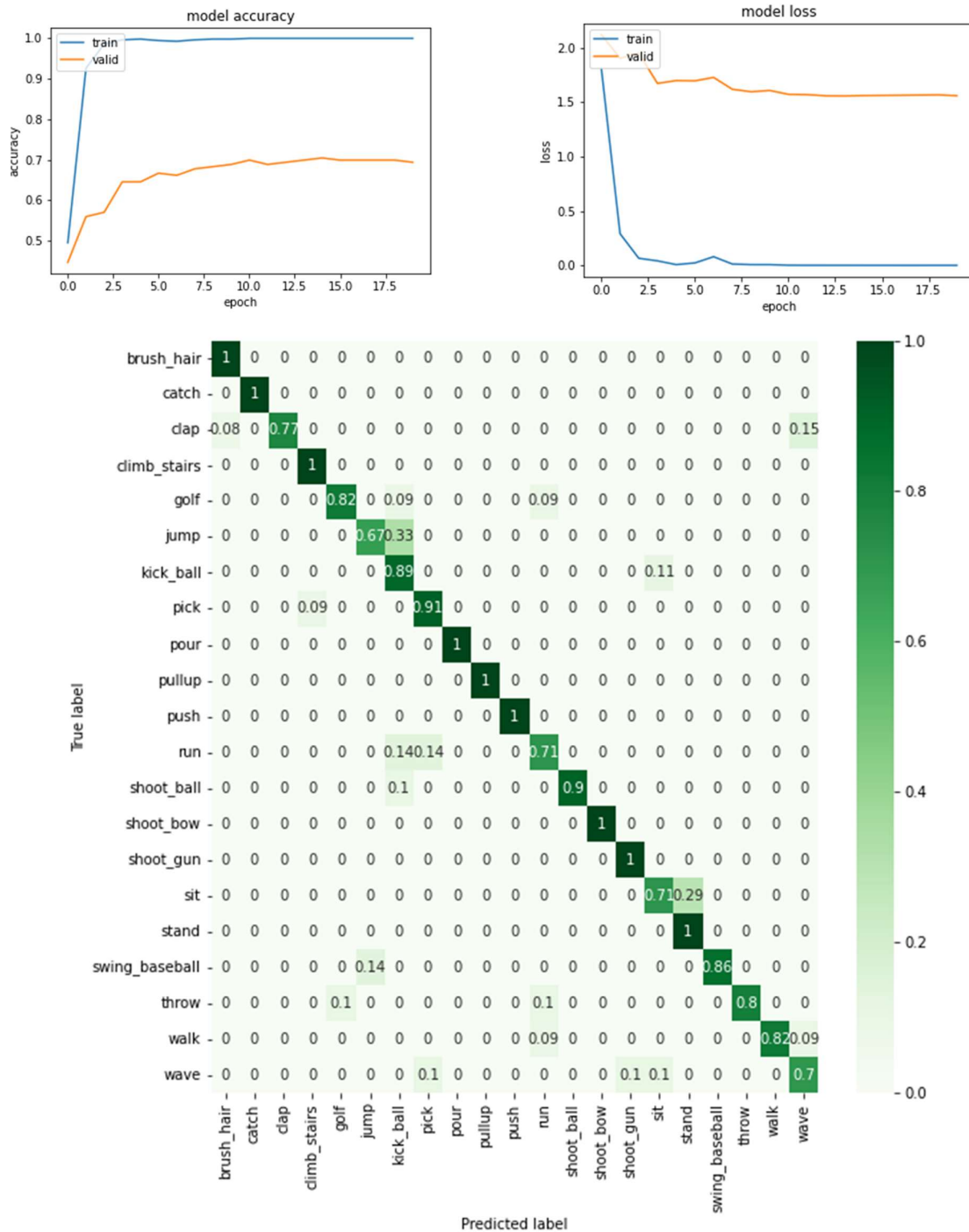


Fig 13: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – ResNet50, JHMDB dataset

DenseNet121 CNN

Among all the models the best results were obtained for when a 2-layer Bi-LSTM with 256 neurons in each layer was used. The accuracy was around 73.2% while the loss was 1.03.

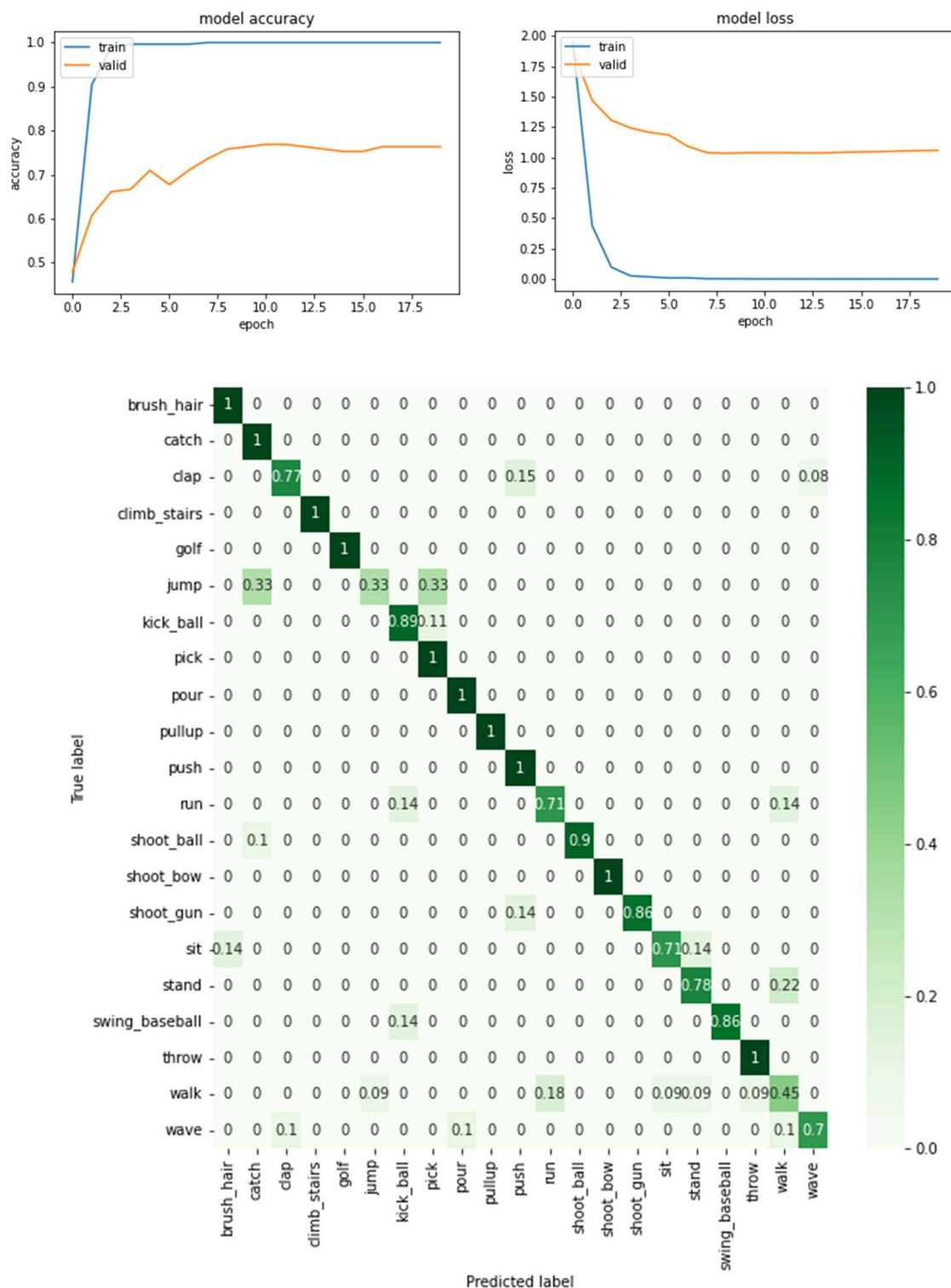


Fig 14: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – DenseNet121, JHMDB dataset

MobileNetV2 CNN

Among all the models the best results were obtained for when I used a 2-layer Bi-LSTM with 256 neurons in each layer was used. The accuracy was around 73.7% while the loss was 1.17.

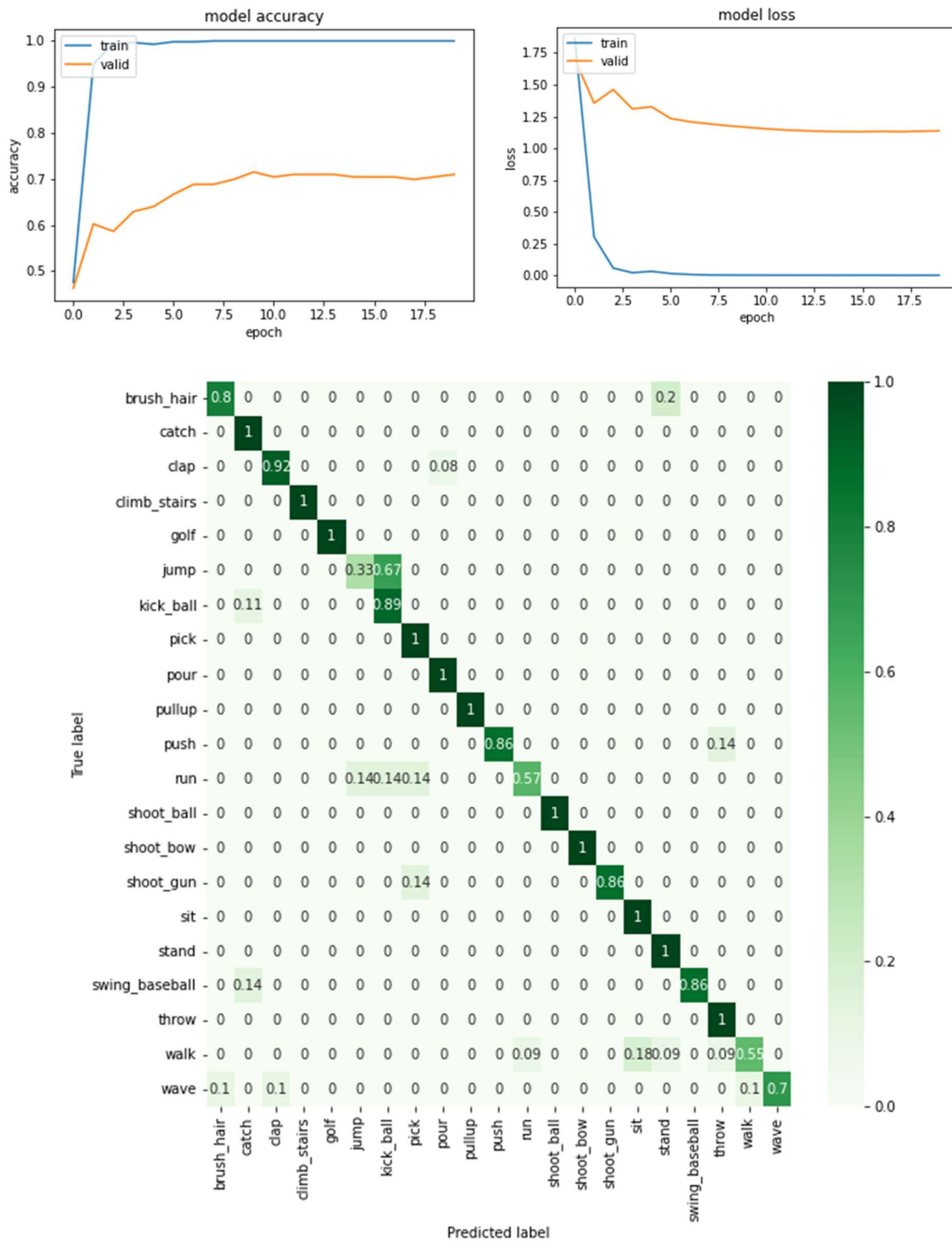


Fig 15: (Top Left) Model Accuracy, (Top Right) Model Loss, (Bottom) Confusion Matrix – MobileNetV2, JHMDB dataset

8. ANALYSIS

In ResNet50 after applying the pooling layer we get a 1D array of size 2048 for each frame in a video, while for DenseNet121 we get an 1D array of size 1024 and for MobileNetV2 we get a 1D array of size 1280. From this we can guess that among the three ResNet50 should get the most benefits from the Self Attention Model, since the array it generates is way longer than the other two. This can be seen from the results as well, in all three datasets we get a better result for ResNet50 when a Self Attention based Bi-LSTM model was used, whereas for the other two with smaller array a simple Bi-LSTM model performed better.

In general, larger number of neurons extracts more features, whereas deeper layers extract richer features, but if there aren't many features to extract from the given data, number of neurons should be lessened, and for simpler data, lesser layers are suitable.

The results seem to follow this idea. In case of the UCF11 dataset all three best performing models have either 32 or 64 neurons each layer, whereas in case of UCFSports and JHMDB dataset the best performing models have either 128 or 256 neurons each layer.

In case of UCF11 the tasks seem to be vastly different, hence making distinction among them is an easier task and thus requires lesser number features. In UCFSports there are many tasks which have overlapping motions, thus distinguishing them would require more neurons. Finally, in JHMDB dataset the sheer number of different tasks coupled with many of the tasks having overlapping motions make it necessary to have high number of neurons.

For most cases the only consistent property among all of them seem to be the depth, i.e., the number of layers in a model, at either 1 or 2.

For the UCF11 dataset one common mistake which two of the three models made is when trying to identify walking, this can be explained as the motion of walking is similar to a soccer juggling, as in both the cases the person moves their legs in almost similar manner, and also to golf swing, this maybe when the model confuses the golf stick with a leg.

As expected in the UCFSports datasets as well, the most common mistakes were made while identifying golf swing, running and walking. Here the models often classified golf swing and running as kicking.

In case of JHMDB dataset also similar problems plagued all the models, inability to properly classify the running and walking videos and classifying them to other activities such as jumping and kicking. Another class which suffered seriously was the jumping class, getting misclassified as kicking, catching and picking.

For the **UCF11 dataset** the best results were achieved when using **DenseNet121** CNN. For the **UCFSports** dataset the best results were achieved when using the **MobileNetV2**, **DenseNet121** had the same accuracy as well but a bit higher loss. For the **JHMDB** dataset the best results were achieved when using the **MobileNetV2** CNN.

Overall, we can see that being the largest dataset, by a huge margin, UCF11 did indeed give better results as compared to the other two.

9. SUMMARY

Here is a summary of the all the models and their performances.

1. UCF11

Pretrained CNN	LSTM Model	Depth	Neurons (each layer)	Model Accuracy (in %)	Model Loss
ResNet50	SelfAtt. Bi-LSTM	2	32	99.1	0.035
DenseNet121	Bi-LSTM	1	32	99.3	0.037
MobileNetV2	Bi-LSTM	2	64	97.8	0.077

2. UCFSports

Pretrained CNN	LSTM Model	Depth	Neurons (each layer)	Model Accuracy (in %)	Model Loss
ResNet50	SelfAtt. Bi-LSTM	3	128	83.3	1.24
DenseNet121	Bi-LSTM	1	256	89.9	0.57
MobileNetV2	Bi-LSTM	2	128	89.9	0.52

3. JHMDB

Pretrained CNN	LSTM Model	Depth	Neurons (each layer)	Model Accuracy (in %)	Model Loss
ResNet50	SelfAtt. Bi-LSTM	1	128	73.1	1.43
DenseNet121	Bi-LSTM	2	256	73.2	1.03
MobileNetV2	Bi-LSTM	2	256	73.7	1.17

10.CONCLUSION

In this project I tried to implement a model capable of generating the State of the Art accuracies for three of the most popular small datasets, i.e., UCF11, UCFSports and JHMDB. This was achieved using some pre-trained CNN models and constructing a Self Attention based Bi-LSTM model to process the data.

Some of the end results I achieved were very close to the State of the Art values, with minor variations. For the others the State of the Art values were achieved by implementing other more sophisticated methods and ideas which is out of the scope for this project.

Finally I would like to add that in some of the experiments there is still scope of some minor improvement by adjusting the hyperparameters, which I would look into in the future along with trying out and implementing other developments and improvements in this field.

Even with all the latest developments in this field a lot is yet to be done. One major problem that we still need to overcome is the requirement of huge training dataset. Computer vision is one branch of machine learning which arguably requires the largest amount of data for training as compared to other branches. Generating such huge datasets is no easy task. As for the model, these still require a very capable machine to run them, a lot of works still needs to be put into making them simple enough that even low power devices can run them since mobility would be a big defining feature in the foreseeable future.

11. REFERENCES

- [1] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad and Sung Wook Baik, "Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features".
<https://ieeexplore.ieee.org/document/8121994>
- [2] Khan Muhammad, Mustaqeem, Amin Ullah, Ali Shariq Imran, Muhammad Sajjad, Mustafa Servet Kiran, Giovanna Sannino and Victor Hugo C. de Albuquerque, "Human action recognition using attention based LSTM network with dilated CNN features".
https://www.researchgate.net/publication/352723901_Human_action_recognition_using_attention_based_LSTM_network_with_dilated_CNN_features
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition".
https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
- [4] Gao Huang, Zhuang Liu, Laurens van der Maaten and Kilian Weinberger, "Densely Connected Convolutional Networks".
https://www.researchgate.net/publication/306885833_Densely_Connected_Convolutional_Networks
- [5] Andrew G. Howard, Menglong Zhu, Bo Chen and Dmitry Kalenichenko, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications".
https://www.researchgate.net/publication/316184205_MobileNets_Efficient_Convolutional_Neural_Networks_for_Mobile_Vision_Applications
- [6] Sepp Hochreiter and Jürgen Schmidhuber, "Long Short-term Memory".
https://www.researchgate.net/publication/13853244_Long_Short-term_Memory
- [7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin, "Attention Is All You Need".
<https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- [8] Mayank Mishra, "Convolutional Neural Networks, Explained".
<https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>
- [9] Aakash Kaushik, "Understanding ResNet50 architecture". <https://iq.opengenus.org/resnet50-architecture/>
- [10] Pablo Ruiz, "Understanding and visualizing DenseNets". <https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a>
- [11] Imran Junejo and Naveed Ahmed, "Depthwise Separable Convolutional Neural Networks for Pedestrian Attribute Recognition".
https://www.researchgate.net/publication/349337616_Depthwise_Separable_Convolutional_Neural_Networks_for_Pedestrian_Attribute_Recognition
- [12] Strahinja Zivkovic, "RNN – Architectural Types of Different Recurrent Neural Networks".
<https://datahacker.rs/003-rnn-architectural-types-of-different-recurrent-neural-networks/>
- [13] Vincent Rainardi, "Recurrent Neural Network (RNN) and LSTM".
<https://dwbi1.wordpress.com/2021/08/07/recurrent-neural-network-rnn-and-lstm/>
- [14] Gongbo Tang, Mathias Muller, Annette Rios and Rico Sennrich, "Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures".
<https://www.semanticscholar.org/paper/Why-Self-Attention-A-Targeted-Evaluation-of-Neural-Tang-M%C3%BCller/e3ee61f49cd2639c15c8662a45f1d0c2b83a60c1>