B.TECH PROJECT REPORT

On

DESIGN OF CNN FOR MULTI-CLASS CLASSIFICATION OF VIDEOS AND IMAGES

By

Vinesh Katewa, 180001061

Roopraj B S, 180001043

Naman Jain, 180001031

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING,

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

DESIGN OF CNN FOR MULTI-CLASS CLASSIFICATION OF VIDEOS AND IMAGES

A PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

submitted By:

Vinesh Katewa, 180001061

Roopraj B S, 180001043

Naman Jain, 180001031

Discipline of Computer Science and Engineering,

Indian Institute of Technology, Indore

Guided By:

Dr. Aruna Tiwari,

Professor,

Computer Science and Engineering,

IIT Indore

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

CANDIDATE’S DECLARATION

We hereby declare that the project entitled "DESIGN OF CNN FOR MULTI-CLASS CLASSIFICATION OF VIDEOS AND IMAGES" submitted in partial fulfilment for the award of the degree of Bachelor of Technology in Computer Science and Engineering’ completed under the supervision of Dr. Aruna Tiwari, Associate Professor, Computer Science and Engineering, IIT Indore is an authentic work. Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Vinesh Katewa
Roopraj B S
Naman Jain

CERTIFICATE BY BTP GUIDE

It is certified that the above statement made by the students is correct to the best of my knowledge.

Dr. Aruna Tiwari
Professor
Discipline of Computer Science and Engineering
IIT Indore

PREFACE

This report on "Design of CNN for multi-class classisfication of video and images" is prepared under the guidance of Dr. Aruna Tiwari, Professor, Computer Science and Engineering, IIT Indore.

We have attempted to provide a full overview of our approach, design, and implementation of a new method for doing video and picture classification through this report.

We tried to explain the recommended solution as clearly as possible. Analysis of the implemented architecture's ability to improve video/image classification accuracy.

ACKNOWLEDGEMENTS

We wish to thank Dr.Aruna Tiwari for her kind support and valuable advice throughout the project, for giving us a chance at our own pace on our lines, and for providing us with very useful information whenever necessary. We would also like to thank all the faculty members of the Discipline of Computer Science and Engineering for their invaluable support and constructive feedback during the presentations. We would like to express our sincere gratitude to Mr.Rituraj for his guidance throughout the project and for helping us at every stage. Finally, we offer our sincere thanks to everyone else who knowingly or unknowingly helped us complete this project.

Vinesh Katewa
Roopraj B S
Naman Jain

ABSTRACT

Classification of images and action in a video are a very challenging task requiring high computation resources. Image classification in itself is a complex task where multiple convolution and pooling layers extract features from the images and these features go through a set of fully connected layers that classify the features extracted into different classes, this task becomes even more complex and resource heavy with the addition of another dimension i.e. time for video classification problems. Models like VGG, ResNet and NASNetLarge are one of the go-to models for image classification as they provide great results with the complexity of the model that they bring. Using the ResNet and NASNetLarge we proposed a new model for image classification that takes the best from ResNet and NASNetLarge structure, the proposed model is essentially a ResNet50 model with better feature extraction capability from NASNetLarge.

For the task of Video classification we went through a number of models and their combination with each other, SlowFast network is a model that performs 3D convolution on the video frames with different frame rates and combines them to get the class of the input. SlowFast uses ResNet50 model as it's base so we simply used the Modified architecture from image classification here. Another model that we reviewed was VideoCapsuleNet and U-Net, combined these models as well to create a better video classification model. Finally, a SlowFast model implemented in PyTorch with modification from U-Net.

CONTENTS

CHAPTER 1

INTRODUCTION

Before getting into the details of our work, we have provided here the motivation for working on this project and divided the entire work into different objectives.We aim to work on-
1)Video classification-It is the task of presneting or providing a label that is relevant to the video given its frames.
2)Image classification-Image classification is where a model can analyse an image and identify the 'class' the image falls under.For example human,animals.

1.1 MOTIVATION

In the last few decades, there has been an explosion in the requirement for robust image and video classification models due to the enormous expansion in computer power and the amount of video/image data we have. Because video classification is a difficult and time-consuming operation, several attempts have been made to solve it.
In today's environment, video classification models have a wide range of applications. From automatic sorting of goods in production lines to automating video camera surveillance, and so on. We were drawn to this topic since it offered us the opportunity to work on such an intriguing concept with several real-world applications.
We aim to create a solution for the image and video classification problem, and to achieve this we have defined some objectives in the next section.

1.2 OBJECTIVE

a. Modification & improvement of existing algorithms to enhancement the performance

b. *Pre-processing the UCF11 & UCF101 Dataset to be used as input for the algorithm:* In order to check the performance of the algorithm on real life data, we have evaluated the results on UCF11, UCF101, CIFAR10 and CIFAR100 datasets.

c. Efficient and correct classification of Images and Videos

To achieve the objectives defined here, a number of literature review and experiments have been conducted which will be explained in detail in the upcoming section. These will include a description of what the existing solution are in the image and video classification, and how we have tried to come up with a better solution.

CHAPTER 2

LITERATURE REVIEW

This chapter will contain a detailed description of all the existing solutions to the problems we aim to create a solution for. The content will include multiple images and video classification models, their workings, how these models were created, and their architectures. We have used these models in our final work with multiple changes that we tried out in the experimentation chapter that will follow after this chapter.

2.1 VIDEO-CAPSULE-NET

The VideoCapsuleNet from *Kevin Duarte et al.*[1] is a simplified network that detects actions and classes. To learn semantic information for action detection, this network uses 3D convolutions and capsules. The capsules can capture visual and motion features, making action recognition easier. The network features a localization component that pixel-wise localizes actions using the action representation collected by the capsules. The localization network can predict fine pixel-wise segmentation of actions due to the capsules' capacity to learn meaningful representations of actions. It uses a basic encoder-decoder architecture that accepts a video clip as input and outputs the class and pixel-wise localization of actions, reducing the number of network parameters. It is taught from end to end without the need for a region proposal network or optical flow data. A lower-level layer that detects features such as eyes, nose, and mouth, for example, will only send data to a higher-level layer that detects a face. We have seen how the network uses capsules. Let us understand how a capsule network works.

2.1.1 Capsules

Geoffrey Hinton et al.[2] introduced the capsule neural network which is a type of artificial neural network that can be used to better model hierarchical relationships in a machine learning system. The capsule network is used to classify images. A capsule is a collection of neurons that can be used to simulate various entities or sections of entities. A capsule network has several capsules in each layer. A capsule has an orderly means of representing an entity's presence and a 4x4 matrix (position) that learns the relationship between that entity and the viewer. By multiplying its own pose matrix by trainable viewpoint-invariant transformation matrices that might learn to reflect part-whole relationships, a capsule in one layer votes for the pose matrix of many different capsules in the layer above. An assignment coefficient is used to weight each of these votes. The Expectation-Maximization from *Geoffrey E Hinton et al.* [3] technique is used to iteratively update these coefficients for each image, ensuring that the output of each capsule is sent to a capsule in the layer above that receives a cluster of similar votes. This allows the network to create part-to-whole relationships between entities and capsules to learn viewpoint invariant representations. The EM routing algorithm used to cluster the data points is discussed below.

2.1.2 Expectation-Maximization(EM) algorithm

The goal of EM routing from *Geoffrey E Hinton et al.*[3] is to use a clustering technique to create a group of capsules with similar features. We utilise EM clustering in machine learning to group data points into Gaussian distributions. EM routing is used to iteratively determine the posture matrix and activation of the output capsules. With alternate calls between an E-step and an M-step, the EM technique fits data points into a mixture of Gaussian models. Each data point's assignment probability r_{ij} to a parent capsule is determined in the E-step. The M-step recalculates the values of the Gaussian models using r_{ij}. The iteration is repeated three times, with the last output being the parent capsule's output.

Now we'll talk about the UNet architecture, which was used to rebuild the images and videos. Before uploading the image to the VideoCapsuleNet network, we needed to reconstruct it to get more accurate information about it.

2.2 U-NET

The main idea in UNet is introduced by *Olaf Ronneberger et al.*[4] is to combine a regular contract network (convolutional network architecture) with consecutive layers, where pooling operators are substituted by up-sampling operators and the output resolution is increased. High-resolution characteristics from the contracting path are merged with up-sampled outputs to localise. We have a large number of feature channels in the upsampling section, which allow the network to pass context information to higher resolution layers. The network has no fully connected layers and only utilizes a portion of each convolution that is valid. The network comprises a contracting path (on the left) and an expansive path (on the right) (right side). Here we have used the U-Net for the reconstruction of images/ videos by adding a 3-channel output at the end.

2.3 NETWORK ARCHITECTURE

We have seen VideoCapsuleNet and UNet approaches for the classification and reconstruction of images/videos respectively. Now let us understand their network architecture.

2.3.1 VideoCapsuleNet

The video input on the network is $8 \times 112 \times 112$ frames. As shown in the figure 2.1, 8 is the number of frames and 112 is the width and height of pixels of the image passed into the network. The network starts with six 3D convolutional layers i.e $3 \times 3 \times 3$, each containing ReLU activations.This produces 512 feature maps with a dimension of $8 \times 28 \times 28$. There are 32 different types of capsules in the primary capsule layer. The *4x4* capsule's pose matrices and activation are produced using $3 \times 9 \times 9$ convolutional operations and $1 \times 1 \times 1$ strides, and the ReLU and Sigmoid activation functions respectively. A second capsule layer, with a kernel size of $3 \times 5 \times 5$, and strides of $1 \times 2 \times 2$, follows. The second capsule layer is fully connected to the final prediction layer. The network's action prediction is represented by the capsule with the highest activation. The masked class capsule poses are input into a fully connected layer in the decoding part, which yields a *4×8×8* feature map. The location of the action in the video is approximated by this feature map. A sequence of transposed convolutions are then applied to the this output, which produces eight *112×112* location maps. Skip connections from the convolutional capsule layers are employed to ensure that fine positional information is captured in this final localization.

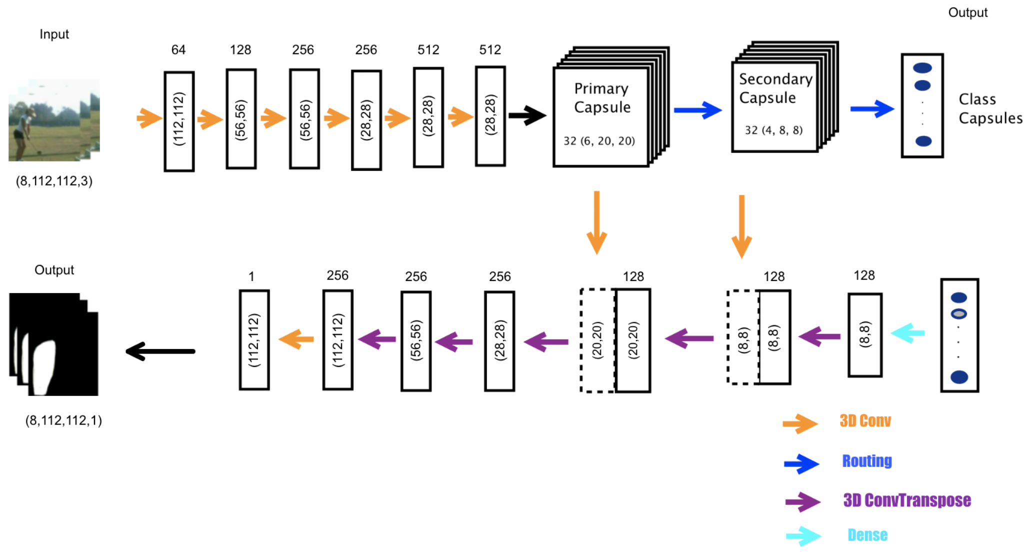

Figure 2.1: VideoCapsuleNet Architecture

2.3.2 UNet

The video input on the network is 8 × 112 × 112 frames. As shown in the figure 2.2, 8 is the number of frames and 112 is the width and height of pixels of the image passed into the network. The provisioning method follows a convolutional network's standard architecture. It is made up of three blocks, each of which has two 3D Convolutions, a rectified linear unit (ReLU), and a max-pooling operation. The middle block has two 3D Convolutions, each followed by a corrected linear unit (ReLU). The expansive path will then have three blocks, each with two 3D Convolutions, a rectified linear unit (ReLU), an up-sampling operator, and concatenation with the contracting path's clipped feature map. Due to the loss of edge pixels in each convolution, cropping is required. To recreate the image at the final level, a 3-channel convolution is utilised to map each 64-component feature vector.

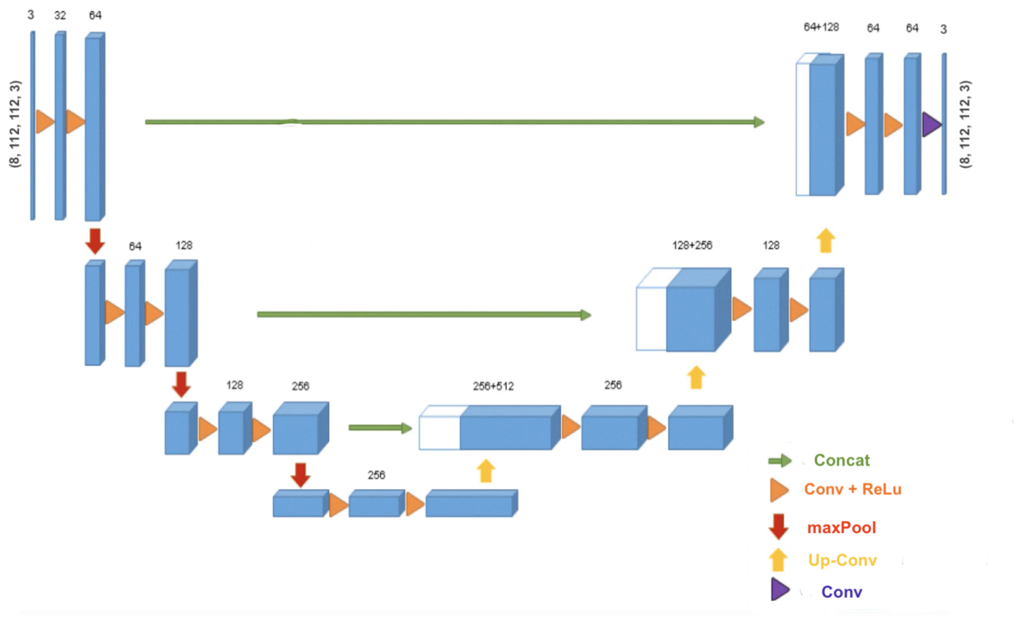

Figure 2.2: UNet Architecture

2.4 SLOWFAST NETWORK-FACEBOOK AI RESEARCH(FAIR)

Facebook AI Research team claims that video clip basically consists of two distinct parts- firstly static areas of frame which do not change or changes at a very slow rate,secondly dynamic areas which indicate the current situation going on. For instance, a video of a bike running on road will include a relatively static road with a dynamic/moving object (bike) moving quickly in the video clip.In an everyday experience of a person lifting something from ground, the ground and other things remain static and dynamic part is the body of person picking up the object. We present SlowFast[5] networks for video recognition. Our model involves (i) "A Slow pathway, operating at low frame rate, to capture spatial semantics" (ii) "A Fast pathway, operating at high frame rate, to capture motion at fine temporal resolution".

The next subsection will follow up the working of SlowFast Network.

2.4.1 Working of SlowFast Network

Both of our Slow and Fast pathways uses a 3D Resnet model that captures several frames at once and runs 3D convolution operations on both of them. The slow pathway uses a temporal stride value of 16 at a 64fps rate which allows 4 frames to be passed for slow pathway. The Fast pathway uses a smaller temporal stride as compared to slow pathway which is set to 2 at a 64fps rate which allows 32 frames to be passed for fast pathway. The fast pathway uses a smaller channel size as compared to slow pathway which is set to 1/8 of chaneel size to make it lightweight.

The next subsection will follow up the Architecture of SlowFast Network.

2.4.2 SlowFast Network Architecture

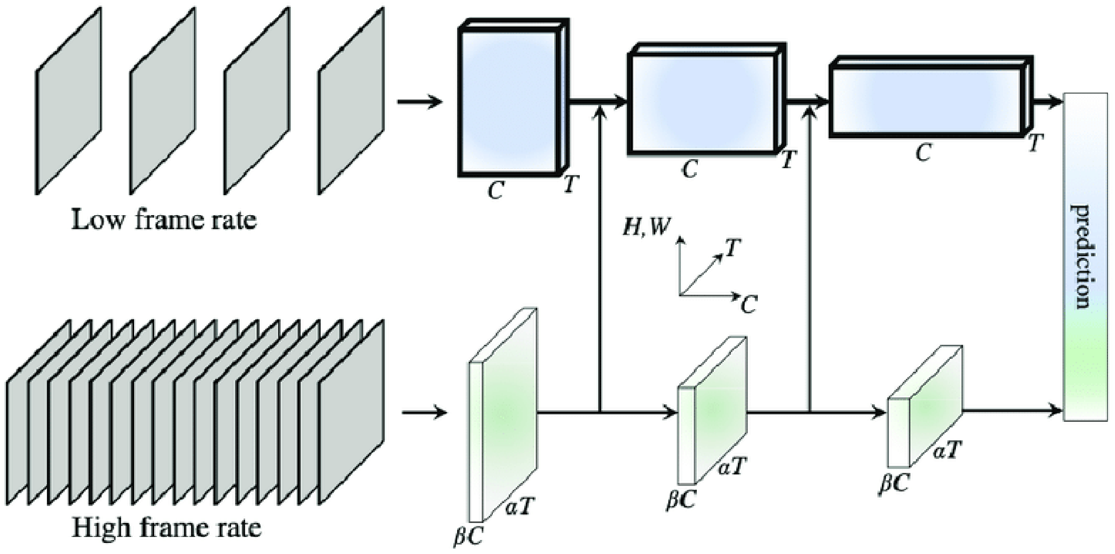

Figure 2.3: The SlowFast[5] Architecture

The following image shows the architecture of SlowFast Network with slow and fast pathways.The middle part shows the 3D convolution operations which uses Resnet50 as backbone and the lateral connections from fast pathway to slow pathway. The orange

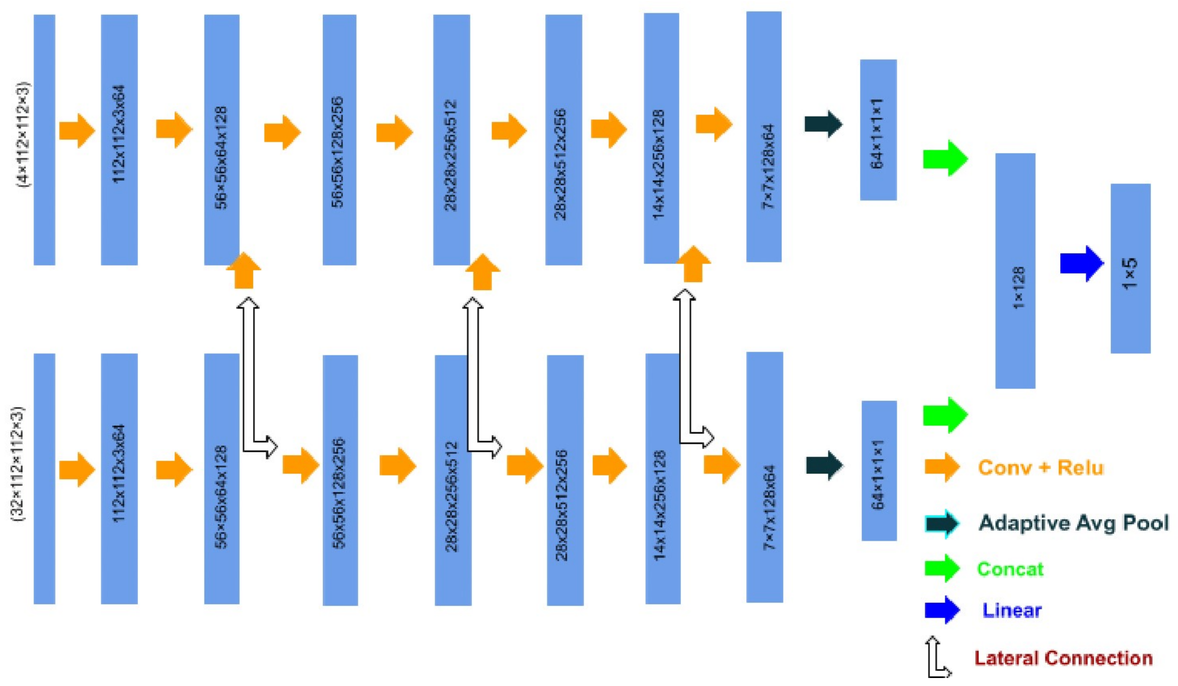

Figure 2.4: Enhanced occ[6] layered architecture

arrows show the 3D convolution operation along with relu activation function and after each of the two 3D convolution+relu operation there is a lateral connection from fast pathway to slow pathway.

At the end of both slow and fast pathways the slowfast perform adaptive average pooling to reduce the output dimensionality and to detect correct kernel size.The slowfast network than concatenates the result from both pathways into a "fully connected classification layer which then uses softmax as an activation function for multiclass classifiation" and action detection in the passed frames.

The next subsection will follow up the Implementation of SlowFast Network.

2.4.3 Implementation of SlowFast Network

SlowFast is implemented in PyTorch. Dataset used for SlowFast network is UCF-11.It contains 11 action categories but for our implementation we have extracted 5 classes from the dataset namely: basketball shooting, biking/cycling, golf swinging, horse riding, soccer juggling.For implementation 3D ResNet model is used in both Slow and Fast pathway. To improve this model, one way that can be done is to use an image classification model (used for fast and slow pathways) that is better than the ResNet. So, the next few sections will cover image classification models.

2.5 VGG NETWORK

VGG[7] networks which are models made with combinations of 2D convolution and pooling layers. The number of layers used in the network reflect within the name of the model, for example, the VGG16 model consists of 16 layers and VGG19 consists of 19 layers. This model was showcased in the ImageNet competition where it reached 92.7 % top-5 accuracy for test data split. The dataset consists of 14 million images with a total of 1000 classes. Instead of using large kernel-sizes, VGG models use smaller and larger number of 3x3 kernel-sized filters in a succession. This makes the model a significant improvement over the existing Image classification models. Starting with an imput size of (224, 224, 3) the VGG model follows with a number of convolution and pooling layers, finally ending the fully connected layers. The fully connected layers contains 4096 channels but the final layers contains the channels that is equivalent to the final number of classes. The VGG19 model has three more convolution layers compared to VGG16 with the remaining configuration remainig the same. Although the VGG19 has higher accuracy than VGG16 the difference in accuracy is very small with a large increase in parameters to be trained for VGG19. Due to this reason, VGG16 is used more often despite being slightly less accurate. VGG16 architecture in figure 2.5 shows the combinations of convolution and pooling layers and final dense layers creating the output:

Although the VGG model has decent accuracy for image classification, there are still models that outperform VGG, one such model is ResNet model. Next section has a detailed description about the ResNet architecture and it’s performance.

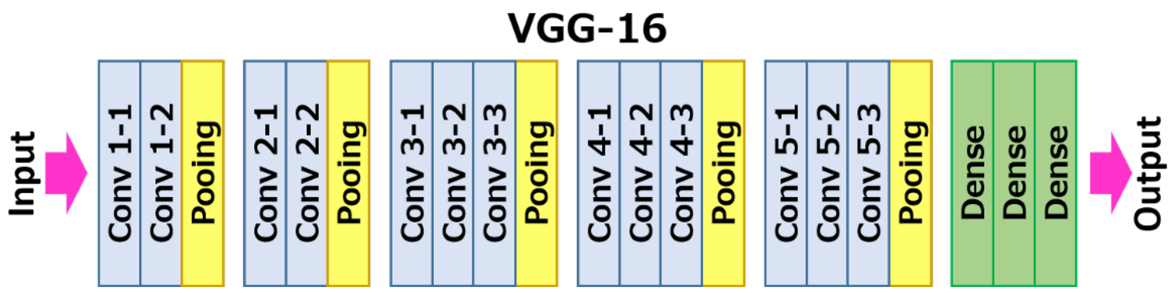

Figure 2.5: VGG-16 architecture

2.6 RESNET50

Before the introduction of ResNet50 architecture He *et al.* [8], the limitation faced by a lot of neural networks was the limit on the number of layers that can be used. This is because of the vanishing gradient problem, with increase in depth of the model the gradient passing through becomes ineffective as it reaches the initial layers. This causes the model to perform worse with the increasing number of layers. To solve this, ResNet50 introduced "Skip connections" to the architecture. These skip connections connected layers which were at least a few layers apart. "These skip connections provided the gradient with an alternate path to flow through during back propagation". This resulted in increasing both the training & testing accuracy of the model. Now, we can create neural network models which contain a large number of layers without having any significant loss in training and testing accuracy. The ResNet model has a lot of variants with different number of layers, the different architectures are named as ResNet50, ResNet50V2, ResNet101, ResNet101V2, ResNet152 and ResNet152V2 with the number within the name representing the number of layers in the architecture. But, during our work we will only be focused on the ResNet50 architecture Simplified view of the ResNet50 architecture can be seen in figure 2.6 in the form on convolution and identity blocks with pooling and final fully connected layers.

Architecture of identity block in figure 2.7 describes the skip connection connecting between three layers directly without any operation in between.

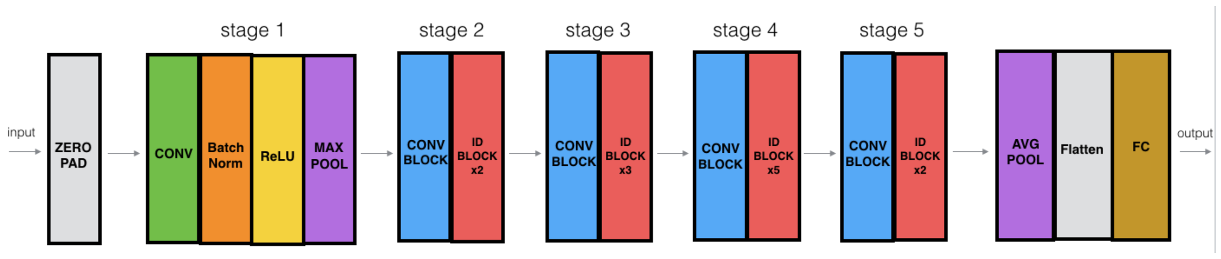

Figure 2.6: ResNet50 architecture

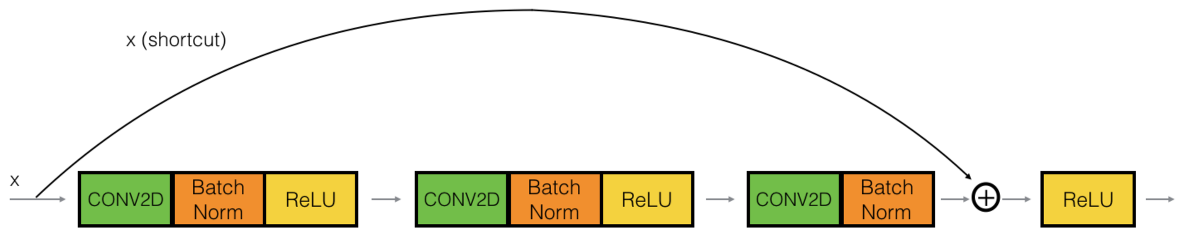

Figure 2.7: ResNet Identity Block

Architecture of convolutional block in figure 2.8 shows the skip connection connecting between three layers apart with a convolution operation on the skip connection.

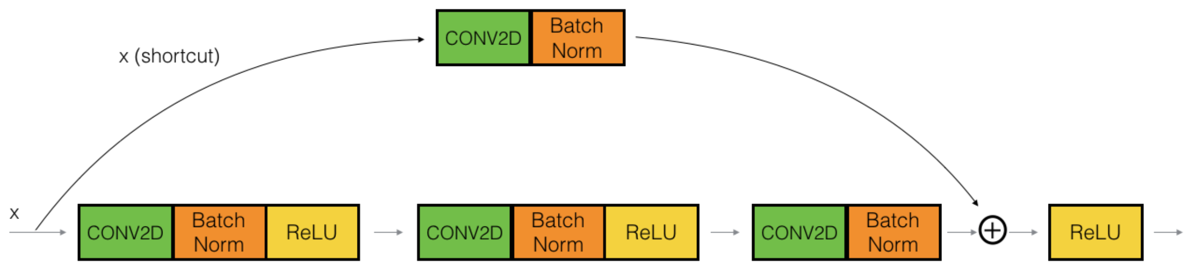

Figure 2.8: ResNet Convolutional Block

In the next section, a new model generated with RNNs named NASNetLarge [9] will be discussed in detail.

2.7 NASNETLARGE

NasNetLarge is a convolutional model where the high level architecture of the model can be seen in figure 2.9, different configuration is used for different datasets depending on the input size.

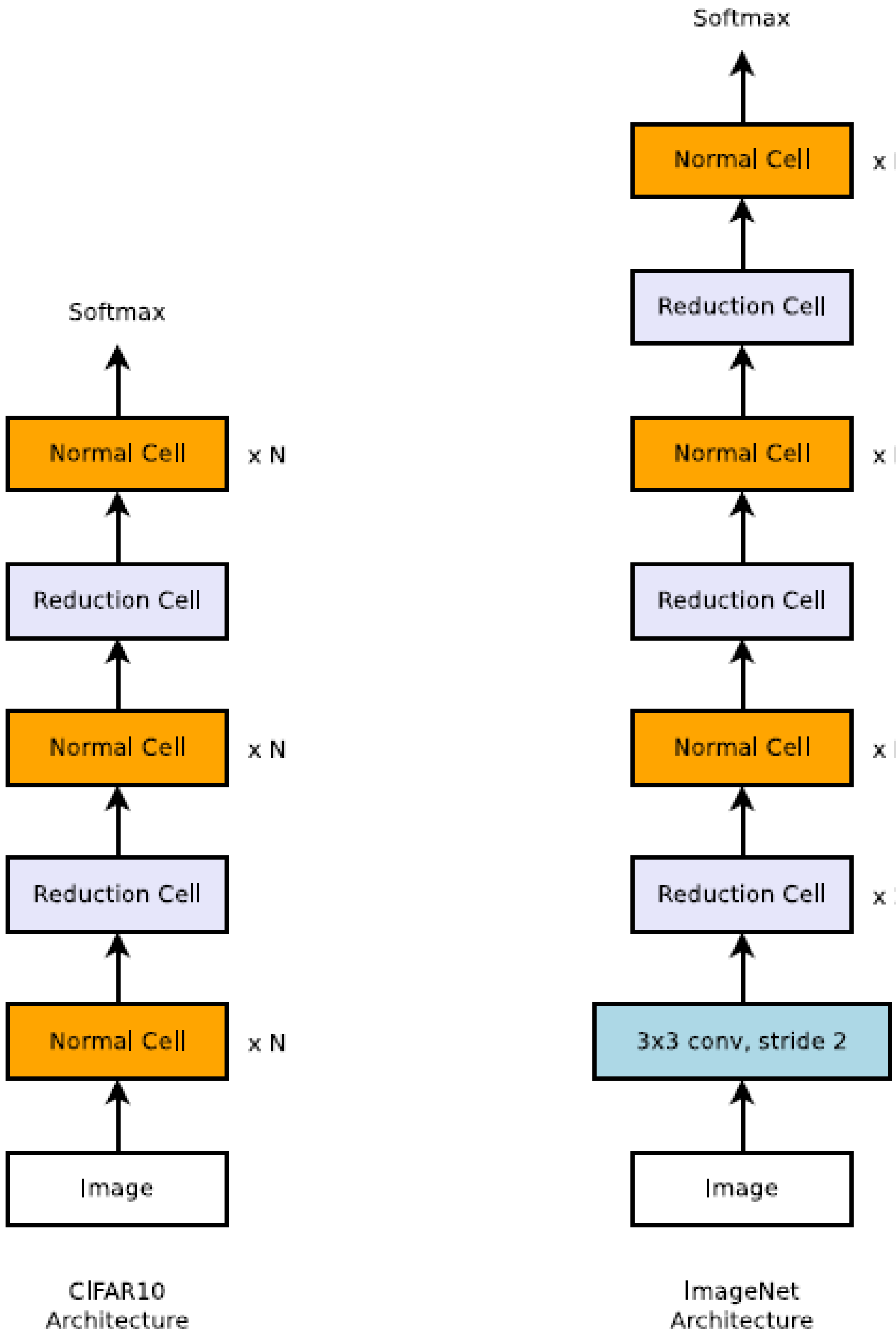

Figure 2.9: NasNetLarge architecture

The overall architecture of the NASNetLarge model by Zoph *et al.* [9] is predefined as a combination of Normal and Reduction cells. However, the architecture of these cells is generated with a Recurrent Neural Network (RNN) model that predicts the combination of different layers as the architecture of these cells, trains the model with the new architecture, evaluates the model and finally creates a new architecture of these cells depending on the accuracy metrics. The architecture for normal and reduction cells can be seen in figure 2.10 where a number of convolution and padding operations is shown that the input goes through before finally combining to create output.

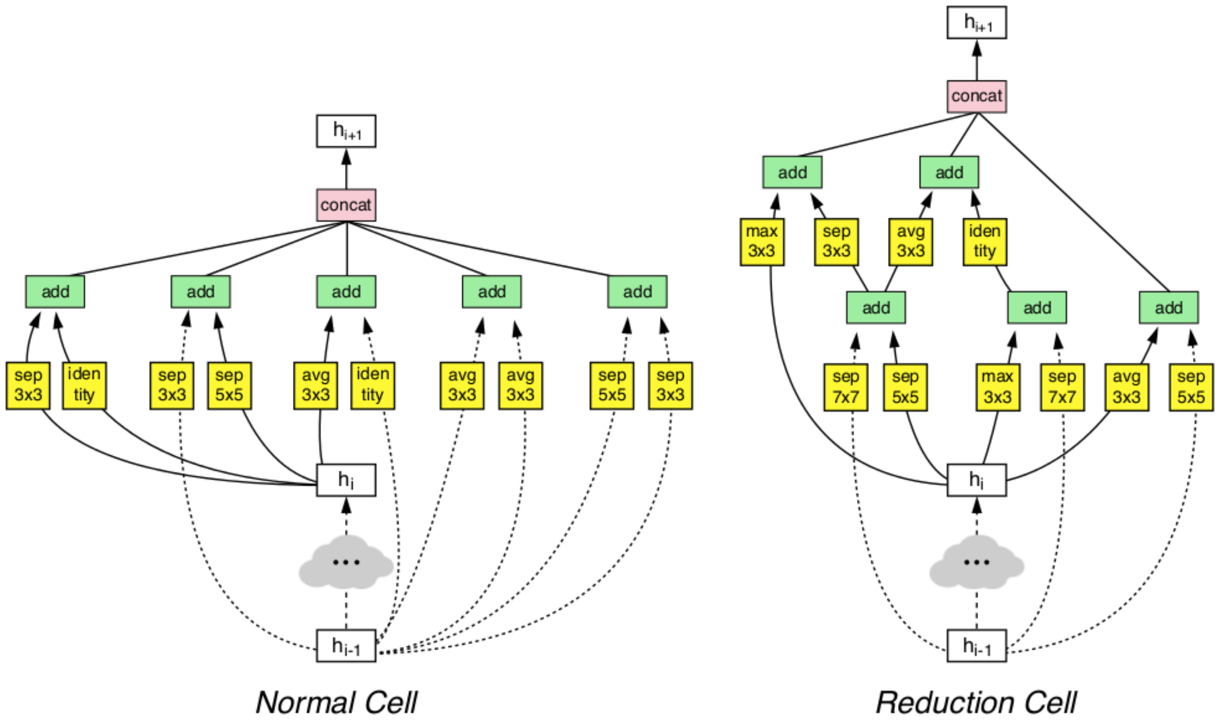

Figure 2.10: Architecture of normal and reduction cells for NASNetLarge

After finishing all the literature reviews that we have done, now moving on to the experimentation done though the project.

CHAPTER 3

EXPERIMENTAL WALKTHROUGH

After the extensive literature review, moving onto the experimentation that we did. In this chapter, we describe the hardware of the systems that we used for training and testing the models, the datasets that the models were trained on, implementations of the existing models and the modifications that we introduced to the models.

3.1 HARDWARE SPECIFICATIONS

A number of different hardwares were used throughout the project.

- The ResNet50 and modified ResNet50 models were both trained using a nvidia GPU, specifically Nvidia 1050Ti Mobile with compute capability: 6.1
- SlowFast model implemented using Keras was trained using the CPU: Intel(R) Xeon(R) Gold 6226R CPU with 16 cores and base frequency of 2.9 GHz.
- VideoCapsuleNet and SlowFast model were both trained using the Google Colab environment
- SlowFast model implemented in PyTorch was trained on Nvidia Tesla K80

To summarize the following systems were used for training and testing the models used through out the project:

1. Nvidia 1050Ti Mobile
2. Intel(R) Xeon(R) Gold 6226R CPU
3. Google Colab
4. Nvidia Tesla K80

Next section describes the datasets that we have used to train and test the models on.

3.2 DATA DESCRIPTION

A total of four different datasets were used for training and testing the models described and proposed during this project, and are described below.

3.2.1 UCF11

Containing a total of 11 different action classes, UCF11 contains over 100 videos per class which were converted into frames for image and action recognition.

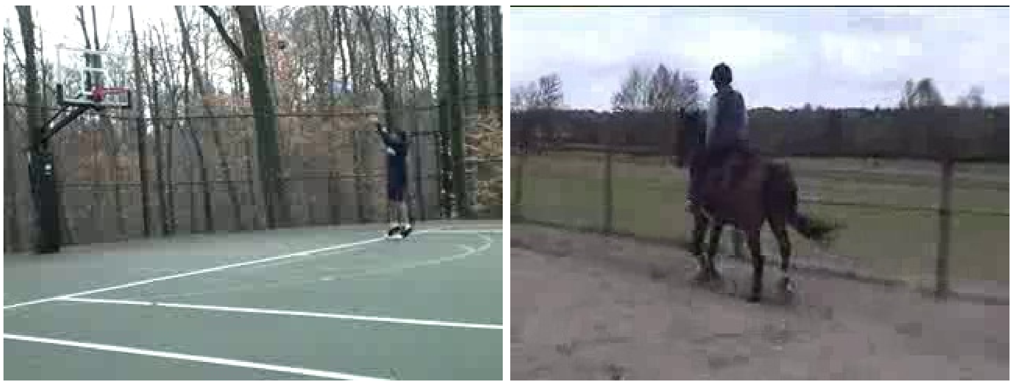

Figure 3.1: Example frames of basketball and horse riding class

Figure 3.1 contains frames from different classes of UCF11, image on the left contains a person playing basketball and another person riding horse in the image on the right. After converting into frames the dataset also became an image classification dataset with over 272,000 different images. Next dataset used was the UCF101.

3.2.2 UCF101

UCF101 dataset from [10] is also an action classification dataset with a total of 101 different classes distributed into 13,320 different videos. Each of the frames from videos was extracted for image classification.

Example frames of archery and playing sitar can be seen in figure 3.2. After converting into frames the dataset became an image classification dataset with over 2,483,000 images belonging to 101 different classes defined. Another dataset that we used is the CIFAR10

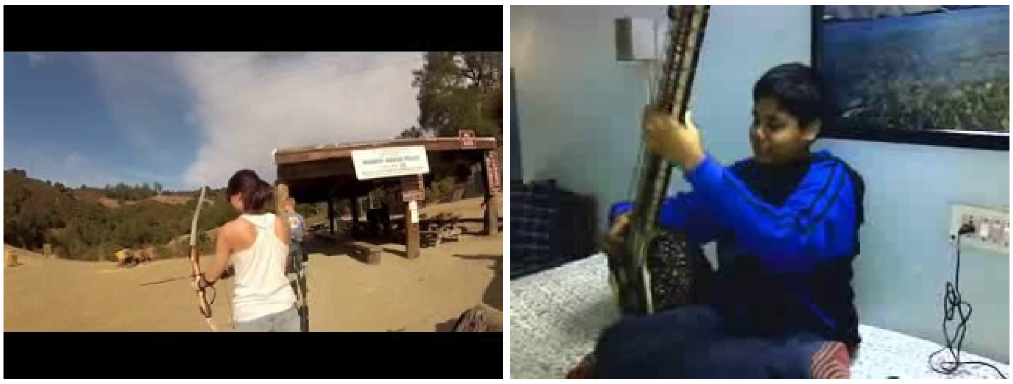

Figure 3.2: Example frames of archery and playing sitar class

described in the next section.

3.2.3 CIFAR10

CIFAR10 is an image classification dataset containing a total of 60,000 color images of size 32x32 divided into 10 classes. The classes included in this dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Example can be seen below:

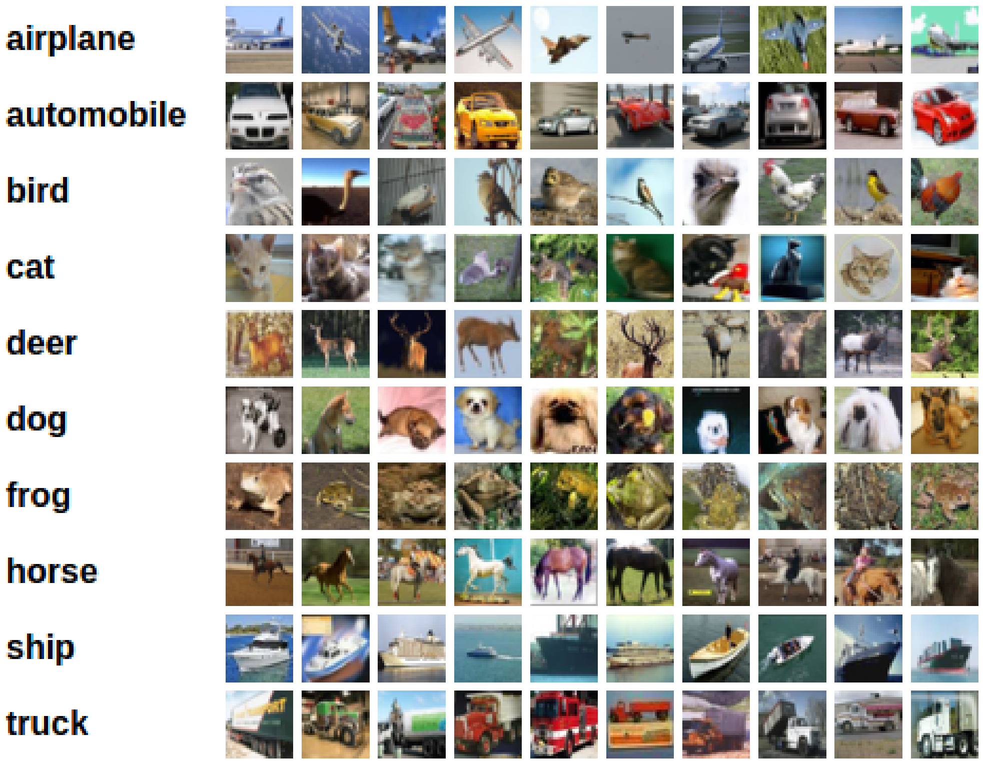

Figure 3.3: Example of images from CIFAR10

Figure 3.3 contains the different classes of CIFAR10 and example images corresponding to each class. CIFAR10 also has a related dataset with a total of 100 classes named CIFAR100, described in the next section.

3.2.4 CIFAR100

CIFAR100 from [11] is an image classification dataset that contains 100 different classes of objects from animals, trees, aquatic animals, foods, vegetables etc. Each class contains a total of 600 color images of size 32x32. Example can be seen below:

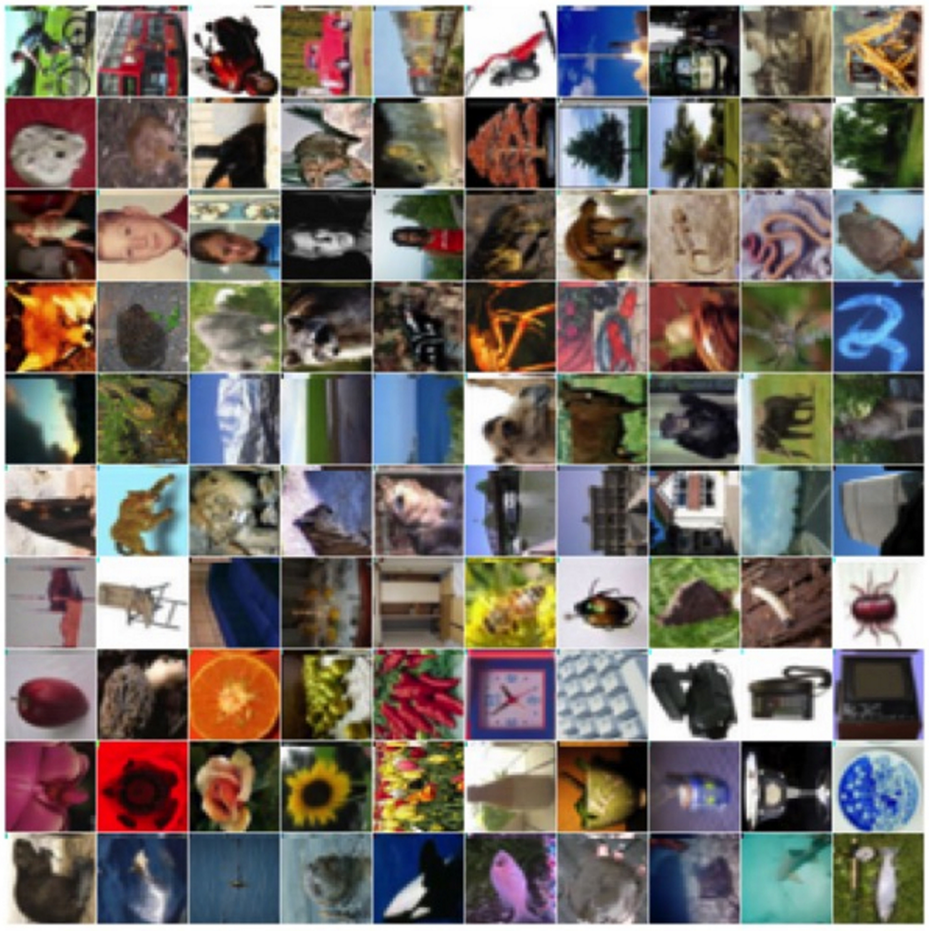
Figure 3.4: Example of images from CIFAR100 dataset

Example of images from CIFAR100 can be seen in figure 3.4. Upcoming sections contain implementations of models and modification over them, starting from VideoCapsuleNet.

3.3 IMPLEMENTATION OF VIDEOCAPSULENET

The VideoCapsNet is implemented using Tensorflow. The network is trained using Adam optimizer, with a learning rate of 0.0001, and batch size is 8. We used UCF11 as a dataset to measure network performance. In the part of video pre-processing, we have downsampled the video data into *160×120* sizes. The ground truth information for all videos was stored in .xgtf format. To read these base truth values, we used Viper software in MATLAB and organized the base truth labeling and bounding box values into a single file.mat file of all video datasets. Then, during training, we cropped *112×112* patches at random from the 8-frame movie, whereas testing frames were cropped from the centre. All pixels within the bounding box are regarded the ground truth foreground, whereas pixels outside the bounding box are considered the background.

3.3.1 Modification Over VideoCapsuleNet

We reduce the convolution layer by removing 2 layers from the encoder part of the VideoCapsuleNet Network as shown in the figure 3.5. We have four convolution layers and three capsule layers in the encoder. Then skip connections are introduced in the Network. The 3DConv1, 3DConv2, and 3DConv3 outputs are combined with 3DconvTranspose5, 3DconvTranspose4, and 3DconvTranspose3 respectively. The high-resolution features are combined with upsampling output from the decoder. This modification helps in the faster convergence of localization components in the network. The Network is trained end to end with an objective function that is combination of the classification and localisation losses. Since the localization components converge faster, this results in reducing the time needed for the training of the network.

3.3.2 Objective Function

The network is trained from end-to-end with an objective function that combines classification and localization loss. For classification, we used spread loss. This is calculated as:

$$L_{\mathrm{c}} = \sum_{i=1} (0, m - (a_{\mathrm{t}} - a_{\mathrm{i}}))^2$$

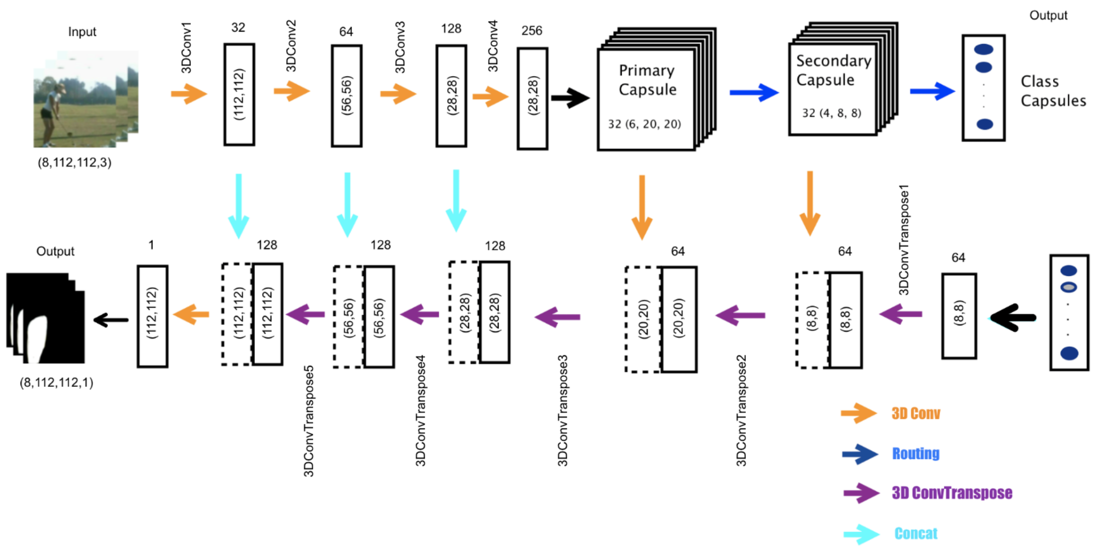

Figure 3.5: Enhanced VideoCapsuleNet

Where a_t is the target class activation and a_i is the activation of the final class capsule corresponding to capsule *i*. During training, the margin *m* is linearly increased from 0.2 to 0.9. We employ sigmoid cross entropy with the expected sets of segmentation maps for action localization to reduce localization loss.

$$L_s = -\frac{1}{TXY}\sum_{k=1}^{T}\sum_{i=1}^{X}\sum_{j=1}^{Y}[\hat{P}_{kij}\log(P_{kij}) + (1-\hat{P}_{kij})\log(1-P_{kij})]$$

Where,

$$P_{kij} = \frac{e^{F_{kij}(\hat{v})}}{1+e^{F_{kij}(\hat{v})}}$$

Is the posterior probability of a pixel in the predicted volume at position *(k,i,j)* for an input video v. F_{kij} is the activation value for a pixel in the anticipated volume of an input video v at location *(k,i,j)*. The network prediction's shape *(T, X, Y)*, where *T* is the temporal length and *X* and *Y* are the height and width of the prediction volume, respectively. Thus VideoCapsuleNet is trained using the objective function

$$L = L_c + \lambda L_s$$

Where λ is used to reduce the localization loss so that it does not overshadow the classification loss.

3.4 IMPLEMENTATION OF UNET

Tensorflow is used to implement the Unet architecture. With a learning rate of 0.0001 and a batch size of 8, the network is trained using Adam optimizer. The UCF11 data set is used to evaluate Unet architecture's performance in image reconstruction.

3.4.1 Objective Function

This is trained end to end using the reconstruction loss. We have used the mean square error loss to train the this model. The loss is the mean overseen data of the squared differences between true and predicted values.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (y - \hat{y}_i)^2$$

where $\hat{y}$ is the predicted value.

3.5 IMPLEMENTATION OF RESNET50

Using the out of the box ResNet50 model predefined and pre-trained, provided by the keras library and training the model on the frames of UCF-11 alternatively named as Youtube action dataset provided very unsatisfactory results.

Because of unsatisfactory results, the code was re-written layer by layer for the model including batch normalization for better classification accuracy. This iteration resulted in fairly good accuracy. Next we take inspiration from NASNetLarge and modify the ResNet model to propose a new one.

3.5.1 Inspiration from NasNetLarge

NasNetLarge introduced normal and reduction cells that were created using recurrent RNNs introducing a high accuracy model with less trainable parameters.

This led to using the normal and reduction cells inside the ResNet50 layer to increase the feature extraction which should lead to higher accuracy, taking in the best features from both NasNetLarge and ResNet50.

3.5.2 Modification Over ResNet50

Added various combinations of normal and reduction cells between the ResNet50 architecture, trying to increase the features extracted from convolution layers. The higher the feature extraction should lead to higher accuracy of the model. And with a little modification in the added normal and reduction cell configuration we were able to receive very good results on UCF11 and UCF101 datasets. Now, for video classification we can directly use this modified architecture in slowfast network which is discussed in the next section.

3.6 IMPLEMENTATION OF SLOWFAST NETWORK IN KERAS

Written the entire slowfast model in keras, using the ResNet50 architecture for both the slow and fast pathways of the network, this implementation gave not very good accuracy on the UCF11 and UCF101 datasets.

3.6.1 Modifications over SlowFast Network

Since slowfast model is using ResNet50 architecture for both the pathways, we can simply take the modified ResNet50 architecture and use it in slowfast to get a new model. To do this we first converted the ResNet50 code from 2D convolution to 3D convolution and added the modified ResNet50 function to the SlowFast code. This gave fairly good results on training and testing dataset splits.

Following these modifications and proposed models, we move towards the Results. There we have showcased the performance of our models and the accuracy metrics that the models scored on different datasets. The results chapter will then be followed by conclusion and references that we have used throughout our work.

CHAPTER 4

RESULTS

In this chapter, we will present the results of training the models on different datasets. Along with training, we also present the testing and validation scores of the models. We have presented all the image and video classification models here, starting from VideoCapsuleNet, U-Net, ResNet50 and so on.

4.1 VIDEOCAPSULENET

Dataset	Training	Validation	Testing
UCF-11	0.8894	0.8718	0.812

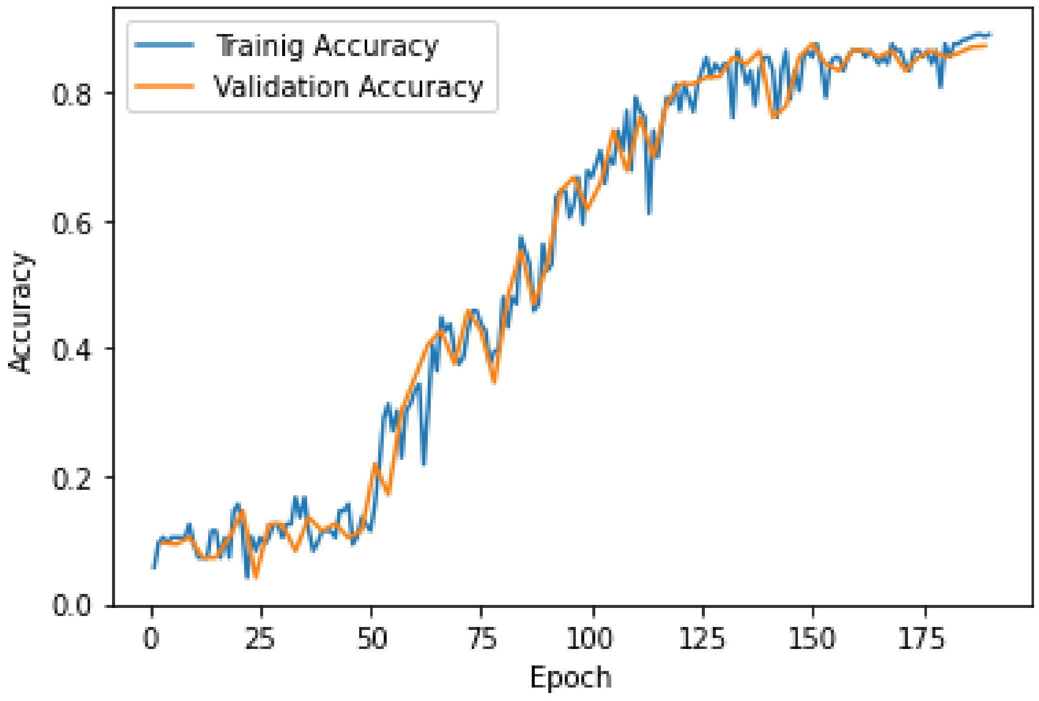

Figure 4.1: The Video-Capsule-Net Training & Validation accuracy

4.2 U-NET

Dataset	Training	Validation	Testing
UCF-11	0.8592	0.7672	0.8480

We have used the PSNR value to check how near the output is to the targeted image/videos. And we could observed that the PSNR values lies the range of 40dB to 50dB.

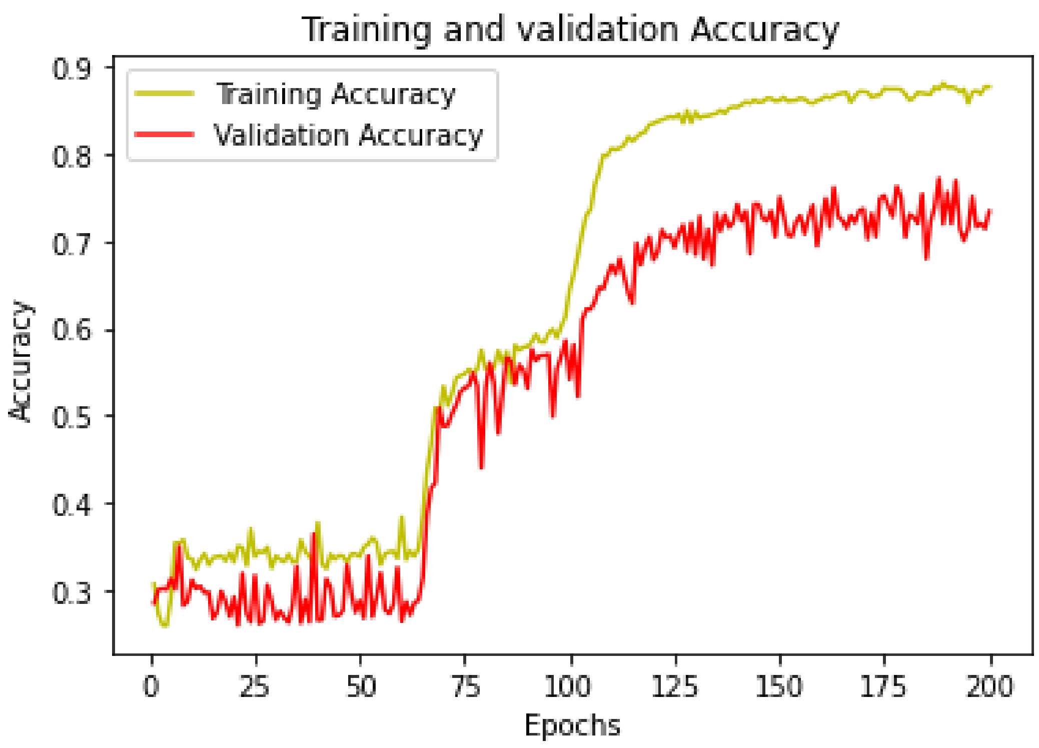

Figure 4.2: The Video-Capsule-Net Training & Validation accuracy

4.3 RESNET50

With building a ResNet50 model from scratch with batch normalization, a fairly average accuracy value can be observed. The accuracy value graph with against epochs can be seen in figure 4.3

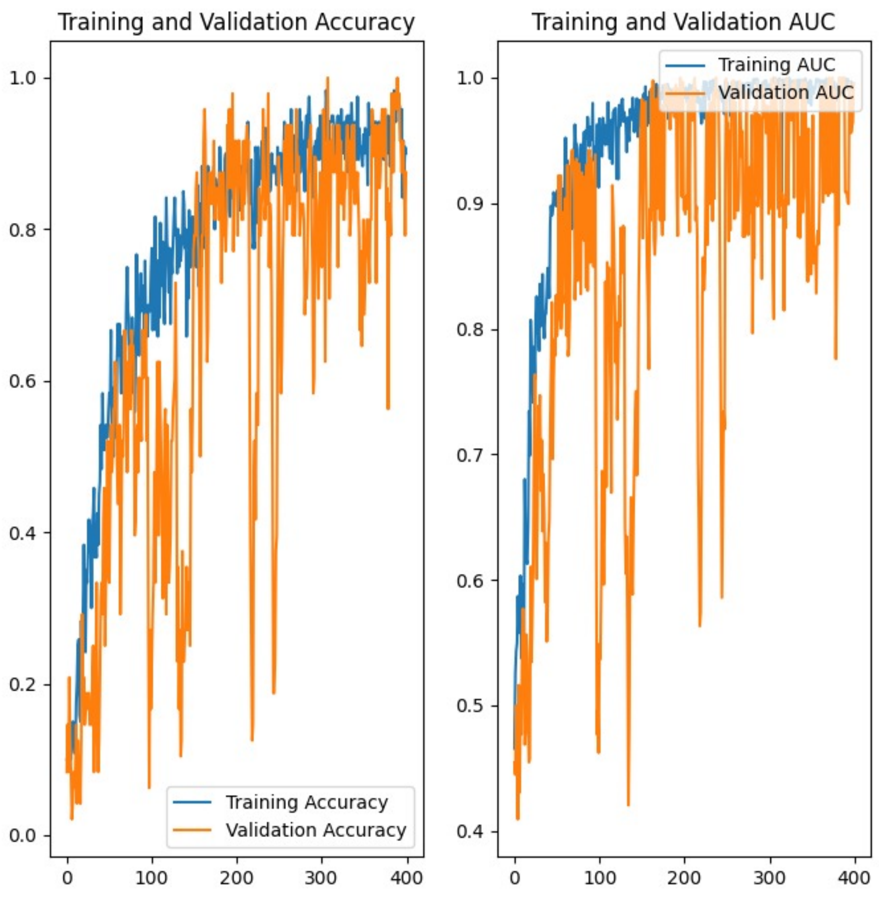

Figure 4.3: ResNet50 Training & Validation accuracy

The results are after training the model on UCF-11 dataset. Further changes included incorporating the normal and reduction cells within the ResNet50 architecture. The architecture can be seen in figure 4.4

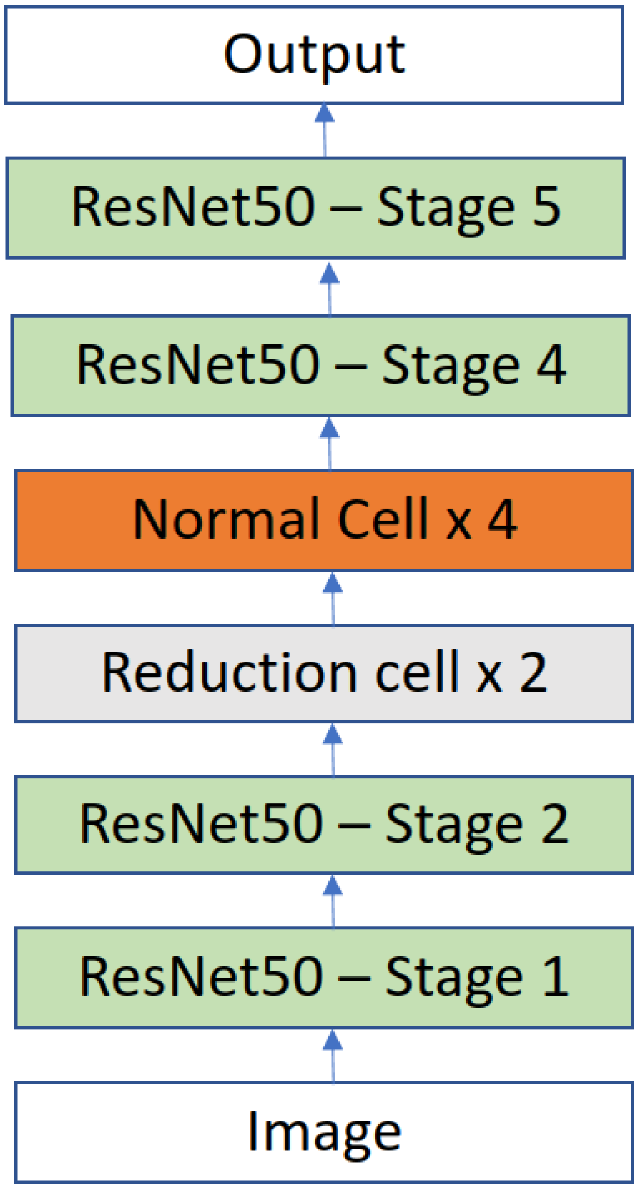

Figure 4.4: Modified ResNet50 Architecture

Training and testing the proposed model on UCF-11 dataset shows the following results in figure 4.5 & figure 4.6, which shows the model reaching high accuracy well over 90%, along with the AUC and top 5% accuracy.

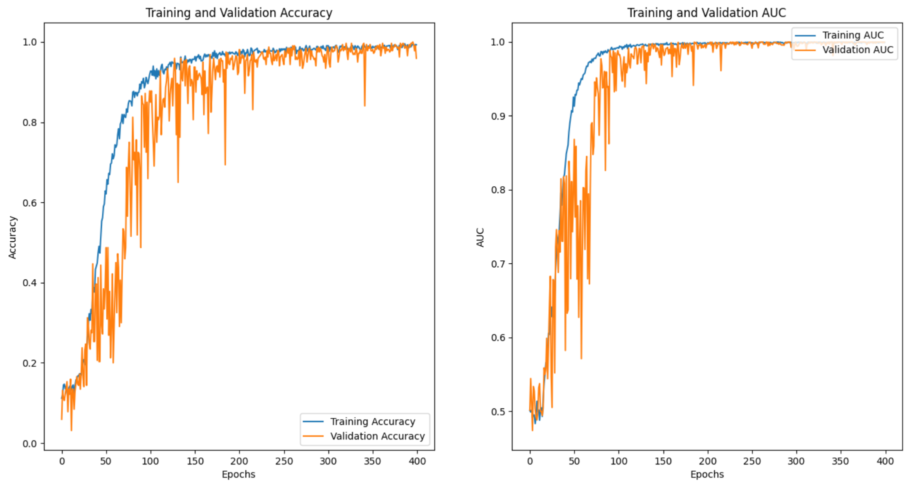

Figure 4.5: Accuracy and AUC graph for UCF11 on modified ResNet50

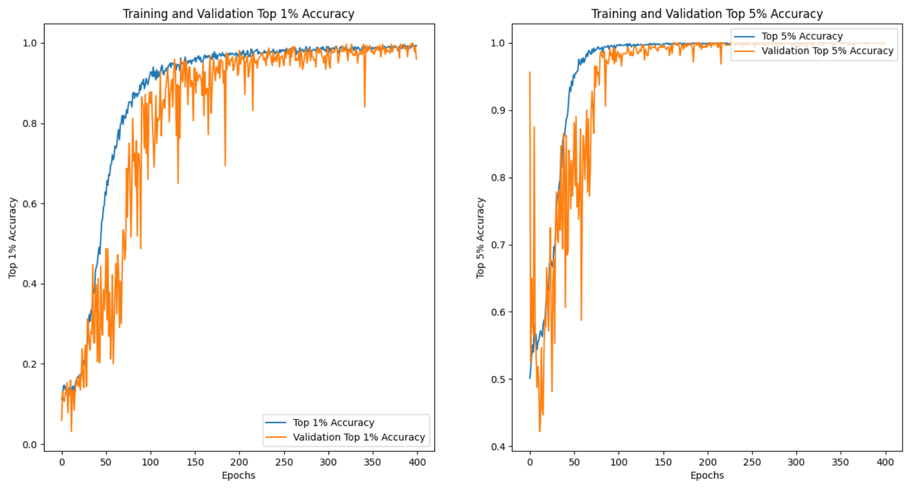

Figure 4.6: Top accuracy fraph for UCF11 on modified ResNet50

The following table shows the different metrics of the trained model on validation dataset, metrics included are time taken, AUC, Top 1% and 5% accuracy and the system used for training and validation.

Time Taken on Test Dataset	556.8 Seconds
AUC	0.9992
Top 1% Accuracy	0.9843
Top 5% Accuracy	0.9998
GPU	Nvidia 1050Ti Mobile

Training and Testing results of the modified model on UCF-101 Dataset can be seen in figures 4.7 and 4.8, containing the graphs for training and validation accuracy, AUC, Top 1% and Top 5% accuracy.

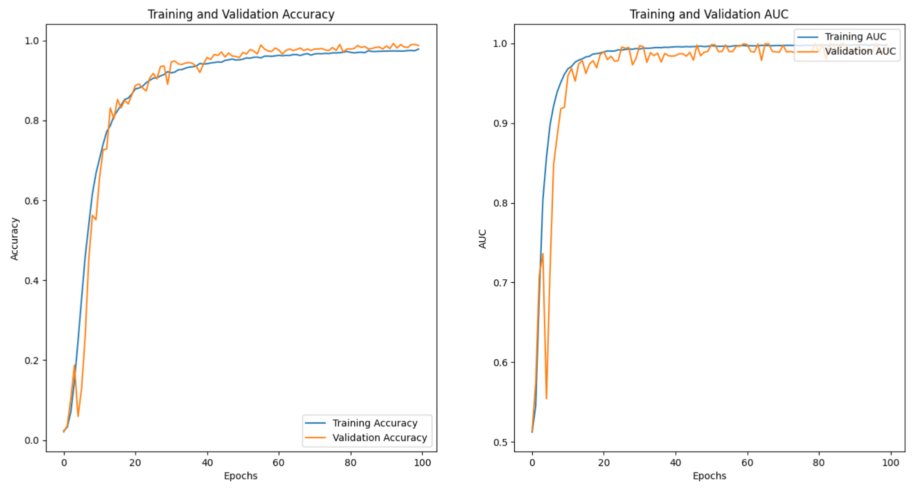

Figure 4.7: Accuracy and AUC graph for UCF101 on modified ResNet50

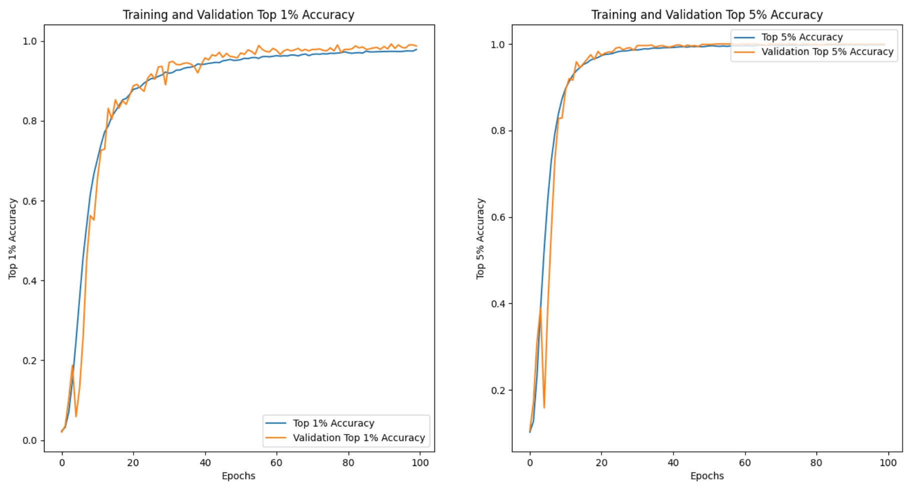

Figure 4.8: Top accuracy graph for UCF101 on modified ResNet50

The following table shows the different metrics of the trained model on validation dataset, metrics included are time taken, AUC, Top 1% and 5% accuracy and the system used for training and validation.

Time Taken on Test Dataset	4177.4 Seconds
AUC	0.9988
Top 1% Accuracy	0.9844
Top 5% Accuracy	0.9988
GPU	Nvidia 1050Ti Mobile

4.4 SLOWFAST NETWORK WITH MODIFIED RESNET50 IN KERAS

Training and Testing results for the modified SlowFast Network on UCF11 dataset can be seen in figure 4.9 followed by the results on validation data in the form of a table.

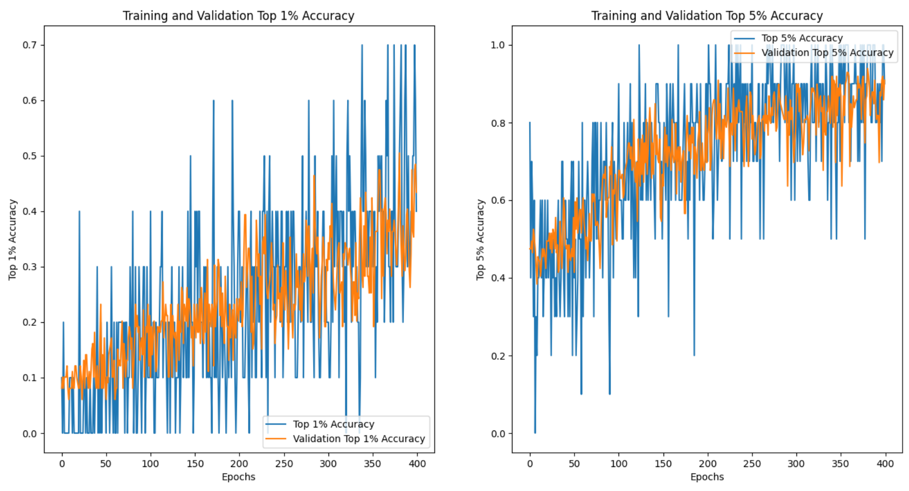

Figure 4.9: Top accuracy graph for UCF11 on modified SlowFast

The following table shows the different metrics of the trained model on validation dataset, metrics included are time taken, AUC, Top 1% and 5% accuracy and the system used for training and validation.

AUC	0.9045
Top 1% Accuracy	0.4343
Top 5% Accuracy	0.9091
CPU	Intel(R) Xeon(R) Gold 6226R

4.5 SLOWFAST NETWORK WITH OCC LAYERS

Dataset	UCF-11(5 classes)
Training	0.6189
Validation	0.4928
GPU	Nvidia Tesla K80

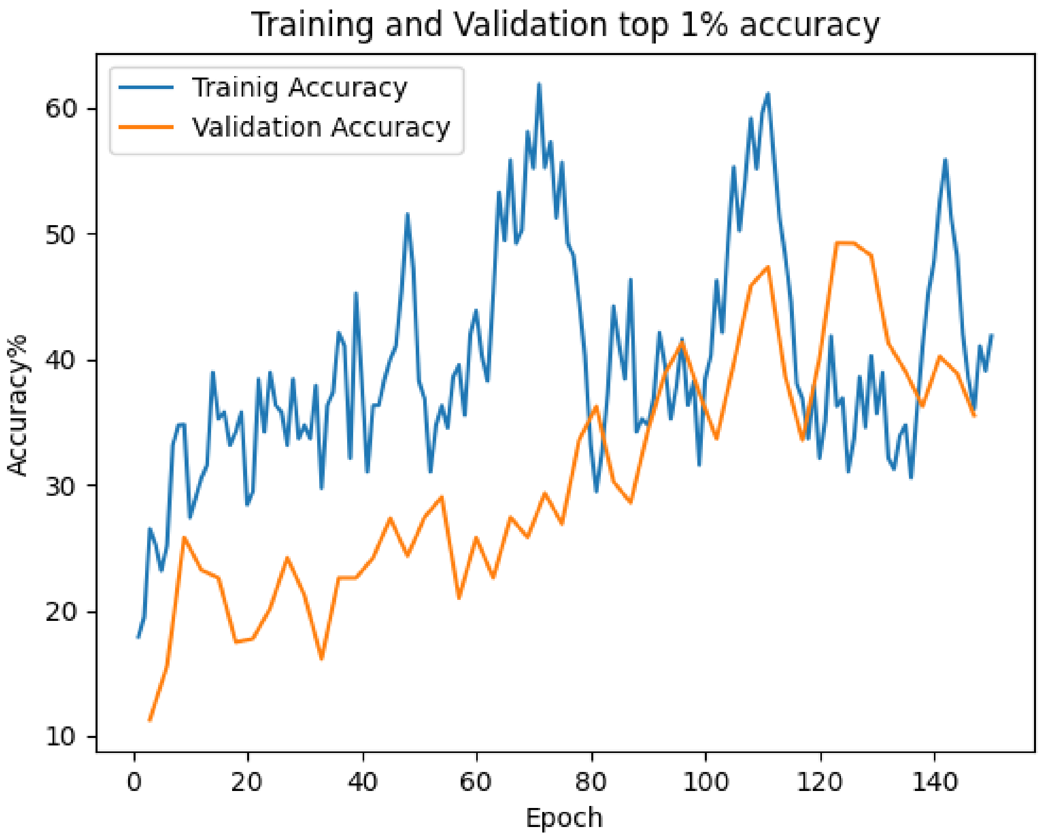

Figure 4.10: Enhanced SlowFast Training & Validation accuracy

CHAPTER 5

CONCLUSION AND FUTURE WORK

We conclude our work by explaining our results for image, action and video classification. How well our models worked for each of the tasks, variation in results depending on different datasets and what future modifications or further testings can be done for the models.

5.1 IMAGE CLASSIFICATION

The proposed architecture for Image classification performed very well on the UCF11 and UCF101 datasets converted to frames, reaching accuracy over 98% on both the datasets. This architecture was also trained on CIFAR10 & CIFAR100 datasets but didn't receive good results because of the time limitations for training, the high accuracy received on the UCF datasets is after training the model over the course of an entire day. With these results we can say that the proposed model is performing very well and needs further training and testing to say confidently that this model is in-fact a robust state of the art model.

5.2 ACTION RECOGNITION

The proposed architecture for action recognition gave better accuracy than the original SlowFast network as proposed by Facebook AI Research. The enhanced architecture with adding additional occ layers and increasing number of lateral connections along with adaptive pooling has given better accuracy than the original architecture by giving an validation accuracy of 49% on our 5 classes of UCF-11 dataset. We also concluded experimentally that giving lateral connection from both the pathways has no impact on our accuracy.

5.3 VIDEO CLASSIFICATION

The modified VideoCapsuleNet for video classification performed well for the UCF11 dataset. The updated network takes video sequence as input and predicts an action class and pixel-by-pixel localization for each frame. This architecture has reached an accuracy of 88%. Also as a part of the reconstruction of the video frames, we used Unet architecture. By substituting the single-channel output with a three-channel output, the Unet architecture was modified to reconstruct video frames. This improved network would accept the picture sequence as input and output the reconstructed image sequence, which would then be merged to create the reconstructed video. With the UCF11 dataset, this network performed well, achieving an accuracy of 85% percent.

REFERENCES

[1] Kevin Duarte, Yogesh Rawat, and Mubarak Shah. Videocapsulenet: A simplified network for action detection. *Advances in neural information processing systems*, 31, 2018.

[2] Jonathan Hui. Understanding matrix capsules with em routing. 2017.

[3] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018.

[4] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. May 2015.

[5] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[6] L. Metz A. Radford and S. Chintala. Deep end-to-end occ. *Unsupervised representation learning with deep convolutional network*, 2015.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[10] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012.

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.